Forging preform design using reverse simulation

Chang, Chao-Cheng

Award date:
2000

Awarding institution:
University of Bath

FORGING PREFORM DESIGN USING REVERSE SIMULATION

Submitted by Chao-Cheng Chang
for the degree of
Doctor of Philosophy
of the University of Bath
2000

Chao-Cheng Chang
16 June 2000

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

Forging preform design, which is a vital pre-requisite for the successful production of forging parts, is still performed extensively by experience-based methods or trial-and-error approaches with or without computer simulations. To improve this time-consuming task and to establish a systematic preform design method, a novel reverse simulation approach is proposed.

The reverse simulation starts from the finished shape with boundary areas being released as dies are moved backwards. The simulation is based on the upper bound theorem with a finite element procedure to achieve, in the first instance, the geometric requirement for complete die filling for two-dimensional plane strain forging. To determine the boundary conditions for each reversing increment, a measure of the material distribution is proposed to enable the setting of a target shape for releasing a contact region from the die surface. The proposed method for controlling boundary conditions ensures that the geometric complexity of the workpiece decreases and thus the reversed shape progresses towards a simple geometry. The procedure has been implemented into the program "Tetrahedral Element Upper Bound Analysis" (TEUBA), being developed at the University of Bath for rapid simulation in two and three dimensions.

Predicted preforms are compared with experimental results or assessed by forward simulations performed by a validated commercial modelling package. The results have shown that the proposed reverse simulation approach has potential in preform design for the forging industry.

Acknowledgement

I wish to acknowledge the financial support of the European Commission in funding the Thematic Network, Forge-Net "Development of modelling tools for the forging industry" to which the research described in this thesis is a contribution, and the Engineering and Physical Science Research Council GR/J61473 (UK) under which the initial TEUBA development work was carried out.

I am very grateful to my supervisor, Professor Alan N. Bramley, for his support, advice and guidance with patience and encouragement throughout this research.

Thanks are also given to Dr. Joe M. Pitt-Francis who wrote a well-constructed code of TEUBA and a development document which helps the work of programming. I am also grateful to the staff in the Department of Mechanical Engineering at the University of Bath for their assistance.

I also wish to thank Dr. Eric Masuku and Mr. Guy Snape for checking grammar and giving suggestions on writing English.

I would like to express my deepest gratitude to my parents who have cultivated in me the desire to learn and provided me with the opportunity to do so.

This thesis is dedicated to my wife, Yu-Kuei, who has supported me in this study from the very beginning. Her encouragement, understanding and assistance have been invaluable contributions.

Contents

List of Figures

List of Tables

Nomenclature

A	Sectional area in two-dimensional plane strain cases
A_e	Sectional area of an element in two-dimensional plane strain cases
a, a^e	Constants
$\mathbf{C}$	Relationship between stress and strain rate
C	Material strength
D	Diameter of cylinder
D_0	Initial diameter of cylinder
E	Number of elements
e	Criterion for terminating backwards tracing scheme
e_e	Criterion for terminating Newton-Raphson iteration
$\mathbf{F}$	Vector of system equations
F	Yield function
F_c	Uniaxial compressive force
F_f	Friction force
F_n	Normal force
f_s	Friction or interface shear stress
G	Plastic potential function
H	Number of nodes in contact with die
h	Height
h_0	Initial height
I_x	Second moment of area about x-axis for workpiece
I_y	Second moment of area about y-axis for workpiece
I_x^r	Second moment of area about x-axis for rectangular target shape
I_y^r	Second moment of area about y-axis for rectangular target shape
$\mathbf{J}$	Jacobian matrix
$\mathbf{J}^{-1}$	Inverse of Jacobian matrix
J_2	Second invariant of deviatoric stress
K	Consistency in Norton-Hoff law
K_0	Constant
k	Shear yield stress in pure shear
L_h	Length of line element
m	Friction factor
N	Number of nodes
n	Sensitivity of strain hardening

n_c	Node in contact with die
n_i^j	Normal vector of node j in contact with die
O	The Big Oh notation
q	Sensitivity of strain rate
S	Boundary surface
S_f	Friction boundary surface
s_{ij}	Deviatoric stress
T	Temperature
T^f	Friction force
T_i^A	Actual collapse traction vector
T_i^U	Traction vector in upper bound theorem
t_o	Time
u_i	Displacement vector
u_i^A	Actual collapse displacement
u_i^U	Displacement in upper bound theorem
$\dot{u}_i$	Velocity vector
$\dot{u}_f$	Tangential velocity vector along friction boundary
$\dot{u}_i^r$	Relative velocity vector along friction boundary
V	Volume for three-dimensional cases or area for two-dimensional cases
$\mathbf{v}$	Velocity at nodes
$\mathbf{v_0}$	Velocity field at time t_0
v_i	Velocity vector
v_i^{die}	Die velocity vector
v_i^j	Velocity vector at node j
v_i^{rj}	Relative velocity vector at node j
w	Number of variables
$\mathbf{X}$	Geometrical configuration
$\mathbf{X_0}$	Geometrical configuration at time t_0
x_1, x_2	Coordinates
Y	Yield stress in simple compression test
Y^U	$\sqrt{\frac{2}{3}}Y$
α	Penalty factor for incompressibility constraints
β	Penalty factor for slide contact friction constraints
Δt	Time increment
δ_{ij}	Kronecker delta
ϵ_{ij}	Strain tensor
ϵ_{ij}^U	Strain tensor in upper bound theorem

$\dot{\epsilon}_v$ Volumetric strain rate
$\dot{\epsilon}$ Strain rate
$\dot{\epsilon}_{ij}$ Strain rate tensor
$\bar{\epsilon}$ Effective strain
$\dot{\bar{\epsilon}}$ Effective strain rate
η Lagrange multiplier
ϕ^i Piecewise linear shape function
λ Positive scalar factor
μ Friction coefficient
Π Total dissipation power
Π^F Friction energy terms
Π^I Incompressibility terms
Π^P Plastic energy terms
Π^S Sliding contact friction constraint terms
ψ^i Piecewise linear shape function
σ Stress
σ_{ij} Stress tensor
σ_n Normal stress
σ_{ij}^A Actual collapse stress tensor
σ_{ij}^U Collapse stress tensor in Upper Bound Theorem
$\bar{\sigma}$ Effective stress
τ_i^f Friction stress

Chapter 1

Introduction

1.1 Forging

Forging is a metal forming operation in which plastic deformation of metal is carried out primarily by compressive forces usually exerted by a hammer or press. The operation uses a set of dies predetermined by a desired product to change the shape of the workpiece without an extensive amount of metal removal [1, 2]. A product forged successfully may also be manufactured economically by other methods, such as machining, casting or powder metallurgy. Advantages of forgings over castings include higher strength, higher overall quality and higher structural integrity without internal defects [3]. In addition, forging deformation controls the metal flow and hence microstructure which is important for durability and fatigue strength in the aerospace industry. In a relative sense, forging products have lower cost and better quality, particularly with regard to strength, toughness, dimensional accuracy, surface finish, and internal or external defects. In general, forging is especially suitable for cases in which: (1) the workpiece geometry is moderately complex, so that costs of machining can be reduced; and

(2) the mechanical and metallurgical properties are important [4]. Applications of forging include automotive components, railroad wheels and jet-engine turbine blades.

Temperature is an important factor in the formability and properties of forged products. According to the workpiece pre-heat temperature, the forging process may be divided into three types, namely, cold forging, warm forging and hot forging. Cold forging is done below the recrystallisation temperature, generally at room temperature, and produces products with good surface finish and dimensional accuracy, which saves material and machining costs. Cold forging may provide products with higher tensile strength than the original material because of strain hardening. The process is applied especially to make round or near round parts, such as gear blanks, in large quantities by the extrusion process. Hot forging is however done above the recrystallisation temperature and requires smaller forging loads, but dimensional accuracy and surface finish are not as good as those of cold forging because of heavy oxidisation. For plain carbon and low alloy steels, a uniform temperature of about 1200°C is required for hot forging. In order to take advantages of both hot and cold forging, warm forging is performed at the intermediate temperatures. Applications that are presently manufactured by warm forging are strongly related to the products made by cold forging, i.e. round or near round parts. Warm forging can be considered as an extension of cold forging to materials that cannot be forged at room temperature because of the demand of costs and quality [5]. The temperature ranges from 200°C to 300°C for aluminium alloys and 550°C to 750°C for steels [6, 7].

The forging process changes the geometry of the workpiece from a simple to a complex shape of the desired product. The process is performed by moving tools to compress a workpiece located between them. The contact area determines the distribution of the applied force on the workpiece. This controls the material flow

to form the desired shape. The geometry of the tools is therefore a critical design factor in the forging process. For a single stage forging process, the tool geometry is determined by the shape of the desired product. The main task in this case is to select the size and shape of the billet and the appropriate manufacturing process conditions that fulfil the requirement of costs and quality. However, in many cases, the finished product is too complicated to be formed by a single forging stage. Intermediate shapes, i.e. preforms, are therefore required to control the metal flow that changes the shape of workpiece to form the final product. In other words, an optimum preform design thus ensures complete die filling, no defects and minimum flash loss and, it may also enable control of the strain distribution and hence microstructure evolution.

One of the key roles in designing preforms is to understand the metal flow during the forging process [9]. This is affected by many parameters including the geometry of dies, the thermal and mechanical properties of the forged metal and dies, the friction at the die-workpiece interface, the heat generation and transfer, and the tool velocity. However, these parameters are difficult to analyse and control. For example, increasing the operation temperature may have the benefit of lower flow stress resulting in a lower forging load, but against this may be the difficulty of controlling the dimensional accuracy of the product. Moreover, a study [10] concluded that the thermo-mechanical stress, induced by the sharp thermal gradient in a hot forging punch process subjected to thermal cracking, is as high as 75% of the total stress field in the area of surface cracking. In this case, the modification of the billet temperature and geometry of dies to reduce the thermo-mechanical stress and stress concentration respectively could be important to prevent cracking or prolong the lifetime of dies. Furthermore, according to a simulation study [11] of temperature and tool velocity effects on forging of blades for jet engines, it has been found that the average workpiece temperature increases by 20 °C and the forging load decreases by 25.2% as the tool velocity

increases from 380 to 625 mm./sec. This can be attributed to the die chilling effects, which are less extensive at higher tool velocity since it takes less time to perform the forging operation. In addition, a high ratio of contact area at tool-workpiece interface in forging of blades, i.e. a geometric factor, also affects the friction and the heat transfer, and hence the workpiece temperature. These interactions between process parameters including the geometry of tools cause the difficulties of analysing and controlling the forging deformation. It is therefore extremely difficult to design preforms from the infinite range of intermediate shapes possible in the deformation sequence of the forging process.

Conventional preform design relies on experience. Empirical guidelines and design handbooks may not suffice for complex forged products. In order to enhance the conventional experience-based preform design methods, computer-aided techniques such as the Finite Element Method (FEM) have been used to predict the metal flow and the occurrence of defects in forged products. Commercial domain FEM simulation packages, such as FORGE2 [12], DEFORM [13] and QForm [14], have provided such information for the evaluation of the forging design process. Although the FEM is able to predict detailed information about the forging process, it may require a high computing overhead in terms of time, equipment and expertise. The FEM combined with design experiences or an iterative optimisation method has been used to design preforms. However, the design procedure is still essentially a process of trial and error, and is still wasteful of time but better than practical trial and error. Moreover, building a knowledge-based system, which captures the existing tool design expertise from industrial experiences or academic research work, is able to offer helpful suggestions. For example, a design system, BID (Blocker Initial-guess Design) [15], constructed by the knowledge-based approach is able to provide and record the information of preform design. Some other techniques such as neural networks and genetic algorithms have also been applied to specific preform designs in which the geometry of the forging

can be classified or represented by parameters. More reviews and discussions on the limitation or drawbacks of current preform design methods are discussed in Chapter 2.

Reflecting on the current solutions of preform design, there is therefore a need to develop a systematic method that can be used to determine directly an acceptable preform design. If such a procedure could be developed, which works fast, then this would further help to reduce lead-time for producing new forged product. As a result, a rapid and approximate Upper Bound Element Technique (UBET) has been widely studied and applied to industrial processes. The technique is based on the upper bound method which is a part of a limit analysis system to predict the minimum energy required to ensure that a material will plastically deform. The UBET has also been developed to run reverse simulations based on the concept of simply reversing the flow simulation by starting from the finished shape with die velocities reversed [16]. However the UBET is constrained to operate only in terms of simple "straight line" profiles whereas this thesis describes an alternative reverse simulation method which is based on the upper bound method combined with the finite element procedure to achieve, in the first instance, the geometric requirement of complete filling for two-dimensional plane strain forging. The proposed idea of preform design has been implemented into the "Tetrahedral Element Upper Bound Analysis" (TEUBA), a program being developed at the University of Bath for rapid simulation of the forging process in two and three dimensions.

1.2 Previous Work of TEUBA

A project entitled "Development of modelling tools for the forging industry" [17] was carried out at the University of Bath to initiate, improve and enhance the application of modelling techniques in the forging industry. One of the objectives of the project was to extend UBET from an axi-symmetric modelling technique to a three-dimensional analysis system which was being developed as TEUBA. The approach based on the upper bound theorem was aimed at maintaining a speed advantage over the finite element method. A brief history of the development of the upper bound theorem for the simulation of forging can be referred to in an article entitled "From Kudo to TEUBA" written by Professor Alan N. Bramley [18].

Before the research presented in this thesis started in September 1996, Dr Joe M. Pitt-Francis had implemented the finite element formulation of the upper bound theorem into TEUBA which was mainly developed for the forward simulation starting from a billet to the finished shape. A basic Graphic User Interface (GUI) with a render area had also been established to load the FE models, set parameters and view the deforming workpiece during the simulation of forging.

1.3 Aim of Study

The aim of this research is to develop a method for preform design using reverse simulation based on the upper bound method combined with the finite element procedure. The study aims to show how a proposed concept of controlling material distribution implemented into the reverse simulation for determining the boundary conditions can be developed to predict preforms for two-dimensional

plane strain forging applications. The research work also attempts to investigate and to improve the numerical treatment in TEUBA. The presentation can be divided into three groups:

(I) Recent Trends of Preform Design

The existing conventional industrial techniques for preform design are outlined first, followed by a survey of the theoretical and experimental methods that have been proposed to design preforms for forging problems.

(II) Upper Bound Method with Finite Element Procedure

The plane strain formulation of the upper bound method with the finite element procedure is summarised and presented. The treatment of the incompressibility of material and the modelling of friction effect are described. Low velocity fields which may cause numerical ill-conditioning are also discussed. Numerical experiments of plane strain compression are carried out to investigate the usability of the modelling of the forging process in TEUBA.

(III) Reverse Simulation with Control of Boundary Conditions

A preform design approach based on the concept of controlling material distribution is proposed. A measure of the material distribution using the second moment of area is presented to deal with the boundary conditions and to evaluate the complexity of the deforming workpiece for reverse simulations in two-dimensional plane strain. The approach aims to ensure that the reverse shape assumes a simple geometry, and provides preforms which achieve, in the first instance, the geometric requirement of complete die fill. The proposed design approach using reverse simulation is then applied to one-stage forging for predicting the billet size, and is also compared with results, especially in terms of deformation trends, from experiments and numerical simulations carried out using commercial software FORGE2. The design method is also applied to multi-

stage forging to predict preforms, followed by verifying the predicted preforms using FORGE2. These results including experiments and numerical simulations show that the proposed design concept for predicting the approximate geometry of a preform has considerable potential for preform design in the forging industry.

1.4 Structure of Thesis

Chapter 2, "Preform Design", presents the experiential design rules of preforms from industry and handbooks, and the current research works on preform design. The task is to review the current development of preform design techniques and hence to understand the limitation of the methods and to point out the difficulties of using computer-aided methods, especially in the reverse simulation approach.

Chapter 3, "Upper Bound Theorem and Forging Modelling", describes the upper bound method for modelling the forging process. The method is part of a limit analysis, and predicts the minimum energy for a large plastic deformation problem at each time increment during the simulation. This sequential limit analysis is then used to model the forging process.

Chapter 4, "Finite Element Implementation", presents the finite element formulation of the upper bound method for two-dimensional plane strain cases. The velocity field is distributed at nodes of the finite element domain to represent the minimum energy of the plastic deformation including the dissipation energy resulting from friction effects.

Chapter 5, "Numerical Treatment and Structure of Computer Code", describes the numerical treatment for solving the system equations. The task also includes the treatment of the ill-conditioning which may be caused by a low velocity field

in the deformation during the calculation.

Chapter 6, "Preform Design Using Reverse Simulation", proposes the new preform design approach using the reverse simulation with a concept of controlling the material distribution. Methods for evaluating the material distribution and for determining the boundary conditions are used to demonstrate the proposed design concept. The method is then implemented into TEUBA to predict preforms.

Chapter 7, "Evaluation of Reverse Simulation for Preform Design", presents and discusses numerical and practical experiments to investigate the proposed design approach. The experiments include the prediction of the billet for one-stage forging and preforms for multi-stage forging.

Chapter 8, "Conclusions", makes conclusions on this study and points out some directions for the future study of preform design using the reverse simulation approach.

Chapter 2

Preform Design

2.1 Industrial Need

To remain competitive, high quality, low costs and short lead-times are always the main concern in industry as new products are being developed. For the forging industry, the quality is measured in terms of the geometric complexity, dimensional accuracy, surface finish, mechanical and metallurgical properties. To achieve the desired quality of forging, potential candidates for material and the appropriate operations must be studied. In the meantime, costs including material, tooling and operation costs are also analysed. However, it is not an easy task to find the optimum solution in a very short time because of the many design variables involved in a forging process.

A forging system comprises many variables including; the geometry and material of both workpiece and tools, the interface conditions between the tools and the workpiece, the mechanics of plastic deformation, the characteristics of the final product, the equipment and the operation environment [4]. The effects of these

variables on different operations may vary. For instance, temperature may have a more significant effect on forging load in a hydraulic press process than in a drop hammer process because of operation time and heat transfer. The strain hardening behaviour of the workpiece may be more important for the quality of products in cold forging than in hot forging because of the flow behaviour and lack of recrystallisation. Therefore, understanding the effects of the variables on metal flow is important to make a successful forging process with high quality and low costs.

According to design experience and handbooks or a knowledge-based selection system [19], it is possible to select an appropriate material and an operation with suitable process conditions for a new product. For example, the recommended forging temperature for aluminum alloy AA 1100 (AA: Aluminum Association) is in the range 315–404 °C [20]. The suggested lubricant for aluminum alloy is graphite [21]. The die material could be AISI H13 (AISI: American Iron and Steel Institute) [22], and manufacturing method for the dies can also be based on existing techniques, such as electro-discharge machining (EDM). The remaining task in developing a new product is therefore to select the billet size and to design the geometry of the dies.

For the final stage of forging, the geometry of dies is highly dependent on the product shape and the design of dies can be helped by the suggestions from handbooks or experience. Preforms, i.e. intermediate shapes, are however extremely difficult to design because of the complex interactions between variables in forging system make metal flow very difficult to predict. Experience of preform design may provide some useful suggestions but it could fail, especially for a new type of product. Computer simulations can provide details of metal flow for examining the quality of products but the tool geometry needs to be determined first but may be modified according to the simulation results. The combination of the

experience and the computer simulation aided approach is however still essentially a process of trial and error and may require a high overhead in terms of computing time, equipment and expertise. An automatic preform design system would therefore be greatly desirable in the forging industry in order to increase quality, minimise costs and to reduce the lead-times.

2.2 Industrial Forging Preform Design

Knowledge of preform design is a key requirement in producing good products and remaining competitive. It is acquired from long experience and is generally kept within the forging company as private property. Moreover, current methods of preform design are too complex to enable systematic rules for general application to be established. This section is therefore only focused on the main principle of the preform design procedure. For designing final stage dies, it is possible to record and systematize the design rules because the die geometry is highly dependent on the desired shape of the product. Therefore, some important factors for designing final stage dies are also presented.

A design procedure starts from discussing the technical specification of the forging component to determine the optimal condition for the manufacturing process. The specification is the basis of the die design, the main design procedure of forging dies is outlined in the following sequence [23, 24]:

1. Prepare ideal forging shape drawing.
2. Design final stage dies.
3. Design preform.

4. Design further preforms if it is necessary.

5. Design associated auxiliary tools.

2.2.1 Ideal Forging Shape

An ideal forging shape is determined by adding some dimensional allowances including machining and die wear to the desired dimensions of the component. Most forging components require machining processes before they are used. It is therefore necessary to consider the magnitude and the application of the tolerances required on a forging to determine the machining allowances on the machined surface. The allowances should be provided at flanges, holes and mating surfaces. To prolong the life of dies, die wear allowances are added to allow for the dimensions of the die being increased after wear. Thermal expansion allowances may also be included because the hot and warm forging processes lead to shrinkage after cooling of the workpiece.

2.2.2 Final Stage Dies

Empirical rules have evolved from experience to aid the design of final dies. These rules suggest appropriate geometric features of dies to control metal flow from a preform to the final shape. The terminology for forging dies is shown in Figure 2.1, and is discussed as follows [25, 26, 27, 28, 29]:

Parting line

During the forging process, metal flows towards cavities and the parting line which is on the plane of separation between the top and bottom dies. The position of

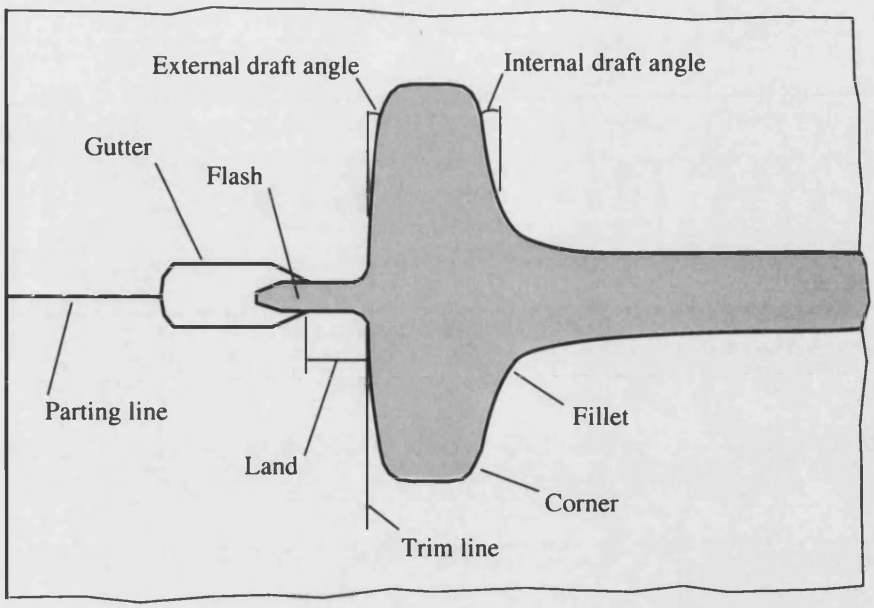


Figure 2.1: Terminology for various features of a typical forging die [28].

the parting line therefore affects the flow pattern and hence can affect forging properties. For most forgings, the parting line is usually placed at the largest cross section of the part. The line may be straight at the centre of the part for simple symmetric shapes or may be irregular and not on a single plane for complex shapes. Moreover, the design of the parting line should also consider the balance of the material flow to avoid an unbalanced horizontal force which will cause a problem of die alignment.

Flash, gutter and land

Some excess flow, i.e. the flash material, should be allowed to flow into a gutter to avoid increasing the forging load unnecessarily. The flash is also designed to allow for extra volume (about 2% to 5%), which gives a tolerance to an inaccurate billet volume, for filling the die impression. This excess volume assures that material will flow across the flash land and thus develop the necessary pressure to assure complete filling of the cavity in detail. The flash clearance between dies is about 3% of the maximum thickness of the forging part. The length of the flash land is about 2 to 5 times the flash clearance.

The gutter is the area in which flash is collected and flow pressure is vented. The gutter must be large enough in width and depth to avoid unnecessary forging load arising from the flash contacting the surface of the gutter.

Draft angles

Draft angles are necessary in forging dies in order to enable removal of the workpiece from the dies. Due to the shrinkage that occurs during cooling in non-isothermal processes, internal draft angles are made larger than external ones. External angles are about 3° to 5° and internal angles are about 7° to 10°.

Fillets and corners

The fillets and corners, which connect surfaces, play an important role in ensuring the smooth flow of the metal in die cavities. Large radii are generally desired to obtain satisfactory metal flows and prevent stress concentrations that would reduce die life. Thus radii should be as large as possible within the constraints of the part features.

2.2.3 Intermediate Operations

A complex shape cannot necessarily be forged without defects and excessive die wear in one operation from the initial billet to the final part. In order to control metal flow, intermediate operations, i.e. preforming stages, become necessary. An important rule in preform design is that the material of the workpiece flows in the direction of the least resistance. Thus, in practical design, the material is generally distributed so that it can completely fill the die cavities [28]. The main design considerations in this stage include [23, 24]:

1. Determination of the number of preform stages

- Stable workpiece location: During the forging process, the deformed workpiece is removed and located into different die cavities. It is necessary to avoid an unstable location for the workpiece which can cause non-uniform die filling.
- Die wear: Friction effects, resulting from the metal flow contacting with the die surface, cause die wear. A preform may reduce the flow over the die surface and thus decrease die wear.
- Internal properties: The internal properties are mainly determined by the flow of the material which is principally affected by the strain rate, temperature and the amount of the deformation. The rate of the deformation may cause the internal temperature of the workpiece to rise beyond a critical value and thus affect the grain size. An appropriate number of preforms is important to control the rate and amount of the deformation and thereby produce the desired internal properties.
- Billet and die fracture: The sufficient clashing area of the die profiles to deal with the impact force is necessary to avoid the die fracture, especially in hammer forging. Sometimes, the rate of the deformation controlled by the stroke speed must also be considered to prevent workpiece fracture.

2. Forging volume and stock size

   The workpiece volume is constant throughout the forging process. This is an important consideration for preform design. The volume is calculated from the full forging envelope including the flash draft and radii from the ideal forging shape. CAD systems may help die designers to calculate this quantity. To determine the dimensions of the billet, the controlling parameters are stock size standards and stable location of the billet. The ratio of diameter and stock length may also need to be taken into account to avoid buckling effects.

Conventional industrial design procedures based on experience cannot provide an accurate analysis for a forging process although an experienced die designer will be able to judge die filling. However, it is clear from the difficulty of predicting metal flow that there are some limitations in establishing design guidelines for new materials and new product shapes by using experiential skills alone. Modelling can be of great help. An approach combining experience with physical modelling or computer simulation may be appropriate.

2.3 Physical Modelling

Metal flow can be simulated by physical modelling which uses a real material to do an experiment for measuring the quality of the design. This approach can be used to obtain the information about the forging process, and hence to verify or modify preforms which might otherwise be designed using empirical rules.

Physical modelling may use a substitute material with similar properties to the desired material to reduce costs or the complexity of the experiment. For example, Cho et al. [30] used the theory of similarity to obtain a model of a hammer and a strain rate sensitive model of a material, using Plasticine, at 21°C. These two models were then used to represent the actual hammer and the actual material AISI 4130 steel at 1200 °C respectively. Ring compression tests with Plasticine were also carried out to model friction effects with a friction factor of 0.3 in which tissue paper was used as a lubricant. This physical modelling method was then applied to verify preforms for a dome-shaped forging product in terms of the capacity of the forging hammer.

Physical modelling can be expensive if a substitute material cannot be found for

modelling the actual forging process, especially where the detailed information is critical; nevertheless, this approach provides a direct method for verifying the design of preforms.

2.4 Numerical Simulation

Numerical simulation considers all the important process variables needed to model real forging processes starting from a billet to the desired product. Simulations using the FEM are able to predict detailed information about metal flow and hence to indicate defects. A computer with the simulation program can therefore be treated as a virtual forging machine for engineers to test preform design before a practical tooling trial is carried out. For instance, Lee et al. [31] used the FEM with thermo-mechanical conditions to investigate four proposed preforms in the bicycle industry. The optimum design, which had a smaller forging load, lower local strain distribution and more uniform temperature distribution, was then selected according to the simulation results. Ward et al. [32] used a commercial FE package, DEFORM, to investigate the multi-stage forge of a railway wheel to establish whether alternative preform configurations of material and die velocities could result in a final component with a low probability of lap defects. It showed that the simulation can provide useful information for a designer in determining optimal tool geometry. The study also considered thermal effects by performing both isothermal and non-isothermal analysis of identical operations, and concluded that thermal effects have a major influence on the results of the simulation of this process. Clearly non-isothermal analysis can be important in hot forging. Other applications of the numerical simulation approach include the comparison of preform shapes of aerofoil sections including circular and elliptical sections [33], and the effect of die deflection on precision forging of aerofoil sec-

tions [34]. These studies have shown that numerical simulation can be used to investigate forging process parameters and hence to help design preforms.

Numerical simulation may offer another approach to verify the preform design but it also requires a trial preform design and the ability to modify the trial preform from the results of the simulation.

2.5 Knowledge-based Systems

A knowledge-based system, i.e. an expert system, can be defined as an intelligent computer program which has the capability to solve problems by using a knowledge base and inference procedures. In general, the system has three principal components: the interface between users and the system, the representation of knowledge and the inference engine [35, 36]. The interface captures user input into the system, and concludes and explains the answer or the suggestion to the user in text or graphical form. The interface may also include an editor to add new information into or change the knowledge of the system. The representation of knowledge stores information in the form of rules in the computer database. The inference engine, which is a computer program and may be constructed on the IF-THEN rules, searches the knowledge database to make conclusions on user inquiry. If the correctness and completeness of the knowledge database can be established, the system with an appropriate inference engine should be able to provide useful information to help the preform design.

Capturing existing tool design expertise from industrial experience or academic research works is the way to build a knowledge-based system. Bakhshi-Jooybari et al. [37] established an intelligent knowledge-based system (IKBS) for the de-

sign of dies and forging processes in this way. The system is able to compare new forging parts with those previously encountered by the system according to the weighting effects of process and geometrical parameters that influence significantly the success of the forging. An experimental method has been developed to obtain the weighting effects for closed-die upsetting and forward extrusion.

Vemuri et al. [15] developed an automatic preform design system, BID (Blocker Initial-guess Design, "Blocker" also means "preform"), using the knowledge-based systems approach. A set of design guidelines gathered from the literature, forging companies and designers were used to construct the system. However, the actual design guidelines were not presented in the article.

Lee et al. [31, 38] developed a computer integrated manufacturing system for cold forging. This knowledge-based system combined with experiential design guidelines and numerical simulations to provide the solution for the die design. A strategy for cold forging process planning was also implemented into the system to assist manufacturing so that all design tasks from product design to actual production could be considered simultaneously.

A knowledge-based system may be suitable for preform design in which the geometry of the desired products can be classified, such as cold forging with axisymmetric shapes. However, for general use, the correctness and completeness of the preform design knowledge is difficult to establish because of the complexity of the forging process in which many design variables are involved.

2.6 Neural Networks

The human brain learns from experience to create internal representations that are used to interpret information from the environment. Scientists and engineers have been studying the function of the learning process and attempting to mimic it by creating artificial neural networks. Neural networks are typically composed of inter-connected units which serve as models of real neurons. Figure 2.2 shows an example of an artificial neural network which is constructed from artificial neuron units as shown in Figure 2.3. A modifiable weight associated with each connection represents the influence between the units. Each unit multiplies each input by the weight on the connection and adds together all these weighted inputs to obtain a total input. The unit then uses an activation function to transfer the total input into an output which becomes an input for each connected unit. Therefore, the behaviour of a neural network depends on both the weights of the connection between units and the input-output activation function. In order to make a neural network that performs some specific task, it is necessary to set the weights on the connections appropriately. This is achieved through a learning process by using an algorithm, such as the back-propagation algorithm, which is designed to compute the weights for minimising the difference between the actual output and desired output. Therefore, the weights represent and store the knowledge of performing the specific task [39].

Training neural networks to predict preforms requires some known design knowledge. Roy et al. [40] used existing knowledge captured from FEM simulations and physical modelling experimental results to train a neural network system with the back-propagation algorithm. The system is able to interpolate within the family of plane strain symmetrical H-shaped components to predict preforms. The components are classified by the ratio of dimensions including inner and outer

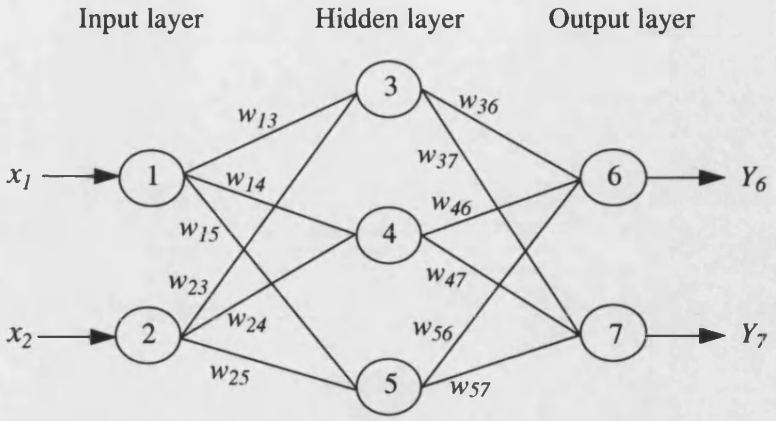


Figure 2.2: Example of an artificial neural network architecture.

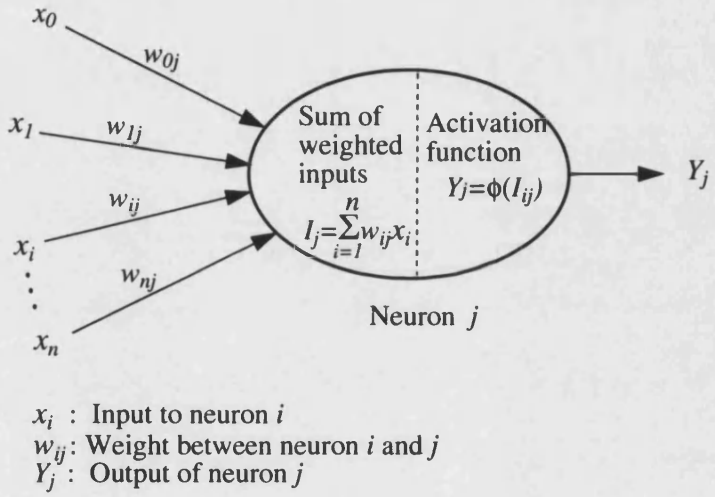


Figure 2.3: Schematic representation of an artificial neuron.

height and width.

Kim et al. [41] proposed a neural network model for function approximation between trials of the initial billet and desired shapes. The selected examples include a plane strain rib-web product in cold forging and an axisymmetric rib-web product in hot forging. The unfilled volume of the FE simulation of the trials was used to train the neural networks system which was constructed by one input neuron for the aspect ratio of the billet, and one output neuron for the unfilled volume. Ten neurons were used for the hidden layer in the study. The

study showed that the neural networks approximate function predicts the unfilled volume differs by about 10% from what was expected from the FE simulation of a new billet size.

Hsu et al. [42] developed a cold forging process design system based on existing knowledge that was acquired by using a finite-element-based program, DEFORM, to simulate pre-defined process condition parameters and tooling geometry. A neural networks model was constructed, in which the multi-layer network and the back-propagation algorithm were used to learn the training examples from the simulation results. The input parameters of the system include workpiece geometry and process conditions, and the output predicts the preform for the axisymmetric cold forging process.

It is apparent from the above that to apply neural networks to the forging preform design, the following elements are essential:

1. A set of parameters to represent the design variables for the inputs to neural networks: The design variables may include geometric parameters such as the size of workpiece or tools, and process conditions.

2. An existing knowledge of preform design to train neural networks: Experiential design rules, experiments or computer simulation results of the forging process help achieve this requirement.

3. An appropriate algorithm to find the weights on the connections of neurons.

Neural networks may provide useful design information for a specific task, in which the shape of products can be classified, e.g. axisymmetric cold forging. For complicated forging shapes, it is however difficult to use few variables to describe the geometry. The existing design knowledge may be limited on a specific

material, a type of shape or a specific forging process. Moreover, using too many layers of neural networks could fail in some cases although it may enforce the system to be memorized and give more accurate answer to the problems related to the original training [43]. These are major drawbacks of neural networks, and should be considered carefully before they are applied to general purpose preform design in forging.

2.7 Genetic Algorithms

The mechanisms that drive natural evolution are not fully understood, but some of their features are known. One of the keys to natural evolution are chromosomes which are organic devices for encoding the structure of living beings. Chromosomes are composed of genes which define the characteristics of chromosomes. In the early 1970's, John Holland and his co-workers were interested in the features of natural evolution, and believed that appropriately incorporated in a computer algorithm, the features of natural evolution might be a technique for solving problems in the way that nature has done — through evolution by means of two primary processes: natural selection and sexual reproduction. Holland and his co-workers began work on algorithms that manipulated strings of binary digits which were called "chromosomes". Owing to its origins in the study of genetics, the field of the study was named *genetic algorithms* [44, 45].

For preform design approaches based on genetic algorithms, the design variables are first represented by chromosomes. Genetic algorithms then solve the problem of finding good chromosomes by manipulating the material in the chromosomes without knowing about the type of forging process. An evaluation of each chromosome is used to select the produced chromosomes so that those with the best

evaluations trend to reproduce more often than those with bad evaluations. Finally, fit chromosomes can be found after many generations, and thus the preform design is solved.

Roy et al. [46] described a method using a genetic algorithm to predict preforms for isothermal multi-stage metal forming processes. The selected examples were multi-pass cold wire drawing, cold drawing of a tubular profile, and cold forging of an axisymmetric automotive component. The design parameters included the die geometry and the number of preforms. The objective functions include the total deformation energy and the difference between maximum and minimum effective plastic strains in the final product, which determines the uniformity of distribution of mechanical properties. The concept of micro genetic algorithms, which uses smaller population sizes, was used in the study to reduce the cost of objective function evaluations using a large deformation FE analysis program. The study demonstrated that genetic algorithms to be a useful technique in the preform design for multi-stage cold forging processes.

Chung and Hwang [47] used a design approach based on genetic algorithms to determine an optimal preform for an axisymmetric product with an H-shaped cross section in a two-stage non-isothermal forging process. The preform geometry was represented by a B-spline curve constructed by 5 design variables which were used to form "chromosomes". The study showed that the approach was able to predict the preform in which the objective function is the measure of non-uniformity of the temperature distribution in the product. The computing time was however high because the number of generations was about 250, and except for the process conditions, the variation of the initial billet size was not taken into account in the study.

In summary, to apply genetic algorithms to the forging preform design, the fol-

lowing components are required:

1. A method to transfer design variables into "chromosomes": A binary string of 1's and 0's could be one of the forms of chromosomes to represent design variables. The design variables may include the preform dimensions and the number of forging stages.

2. A function, i.e. the fitness function, to evaluate the quality of "chromosomes": The total energy consumption, the die filling and the difference between minimum and maximum effective plastic strain in the products could be taken into account to construct the function.

3. A method to produce new "chromosomes": A probability function may be used to generate new chromosomes. Micro genetic algorithms could be applied to control the population size and thus to reduce the computing cost.

For complicated forging shapes, the problem of defining preform geometry by parameters is the same as that encountered in the design approach based on neural networks. Reducing computing cost would also be desirable because the evaluation of genetic strings requires a lot of FE analysis.

2.8 Forward Simulation with Optimisation

The forging process is not only affected by tool profiles, but also by the shape of the initial billet and other process parameters including temperature and friction. The modification of these parameters to obtain the optimal preform design requires a great deal of information about the forming process. To obtain the

information is very expensive and time-consuming when it is done by practical trial and error. A forward simulation of the forging process starting from the billet to the desired shape, which serves as a virtual press, can be combined with an iterative optimisation subject to a special requirement, such as a uniform strain distribution to search for the best preform design.

2.8.1 Achievement of Complete Filling

Complete die filling without folds is a basic requirement for preform design. This requirement can be set as an objective function for an optimisation design approach with the finite element simulation. Fourment et al. [48, 49] introduced a shape optimisation method in which the distance between the simulated shape and the final product shape is set as an objective function to be minimised. The tool profiles are described by spline functions and the design variables are the displacements of the selected characteristic points on the spline in the normal direction. Another optimisation method similar to Fourment and Chenot's method but minimising the difference in area (not distance) between the simulated shape and the desired shape was proposed by Zhao et al. [50, 51, 52]. These two design approaches show that optimisation combined with the FE simulation can predict preforms in which complete die filling is achieved.

2.8.2 Consideration of Quality

The quality of a preform design can be assessed from the mechanical properties which can be affected by strain distribution or homogeneity of the microstructure. Improving die life, such as reducing the stress concentration on the contact surface of dies to avoid cracks, and saving dissipation energy may also be taken into

account for an overall consideration of the quality of a preform design. Kusiak [53] presented an optimisation technique combined with a commercial forging simulation package, FORGE2, to find a tool profile described by a polynomial which gave the most uniform recrystallized grain sizes for an axisymmetric hot forging, in which the grain size was defined as a function of temperature and strain.

Lapovok and Thomson [54] developed a technique to define a possible profile for a preform in which the difference in area between the desired forging shape and the preform was minimised after the main shape parameters were assigned. In this study, the spline function was used to construct the profile of a preform for a plane strain H-shaped forging. The different values of the main shape parameters were chosen according to known constraints in order to define possible preforms. Each preform with a different friction factor was simulated by a finite element package to calculate the ratio of the maximum and minimum equivalent strain used to estimate the degree of homogeneity of strain distribution. After all simulations were performed and the results were analysed, the optimal parameters to describe the preform and the most appropriate friction factor were obtained.

Based on a design similar to the previous one, Lapovok [55] searched for a preform which improved die life by the minimisation of damage accumulation. The damage was described by a function of the plastic strain of die. Six geometric parameters were used to construct forty-seven trial preforms for an H-shape axisymmetric hot forging process performed in three stages. For each trial preform design, the preforming and final stages except the upsetting stage, were simulated by a commercial package, DEFORM. A corner of the die surface in which a high stress appeared was chosen to calculate the damage accumulation for each simulation. Finally the optimal preform was obtained, and the study showed that the die life could be prolonged since the maximum equivalent stress was reduced

from 924 to 597 MPa in the H13 tool steel which was investigated.

2.8.3 Summary on FE Simulation with Optimisation

The optimisation technique with FE Simulation has been used to design preforms subject to such requirements as complete die filling or uniform strain distribution. This approach requires iterative calculations of the FE simulation, and the high computing time is therefore the main obstacle to applying this approach to practical preform design. For cases in which all possible preforms are pre-defined, the optimal preform could be derived from analysing all simulation results. The design procedure with iterative optimisation is briefly summarised as follows:

1. Select main parameters to define the geometry of preform or die profile.
2. Define an objective function for evaluating the optimisation.
3. Determine a trial preform according to known constraints or simulation results. An algorithm is required to ensure that the new trial preform migrates towards the optimum.
4. Use a numerical program, such as an FE package, to simulate forging process.
5. Evaluate the optimisation to determine if a new trial is required.
6. If a new trial is needed, repeat procedure 3 to 5 until the required quality is satisfied.

2.9 Reverse Simulation

The reverse simulation has arisen from the concept of simply reversing the flow simulation by starting from the finished shape with die velocities reversed. With boundary contact areas being released as the die is moved backwards, the simulation predicts an initial billet or a preform when the product cannot be forged in a single forging process.

According to the behaviour of the workpiece, the forging process is a load path dependent process, i.e. the deformation depends on the previous loading history. This means that the forging process is not strictly reversible. An appropriate treatment for modelling a reverse process, which is close to a real forging deformation, is therefore important for using the reverse simulation to predict preforms. This design concept has been investigated by several researchers using different approaches.

2.9.1 Backwards Tracing Scheme

Park, Rebelo and Kobayashi [56] proposed the backwards tracing scheme which was implemented with the finite element method for the determination of preforms in shell nosing. The backwards tracing scheme is an approach for determining the geometry of the workpiece at individual backward increments from the final product shape. A searching algorithm with experiential design rules is used to determine a new trial preform for each backward increment. The trial preform is then repeatedly simulated by a finite element program until the similarity of the current shape and one resulting from the trial preform is satisfied. The scheme is able to predict preforms or its starting geometry, i.e. the billet, if the bound-

ary conditions can be specified at each time increment in the backward tracing procedure. This design approach has also been used for different applications including eliminating crop loss of end shape in a plane strain rolling operation [57], predicting a preform for an aerofoil section blade [58], determining a preform for a net-shape shell nosing component without machining after formation [59], and designing a preform to produce a flat front edge after extrusion [60]. An extension of the backwards tracing scheme to a three dimensional deformation of upsetting a rectangular block was investigated by Kang et al. [61]. These studies have shown that the backwards tracing scheme is a viable preform design approach. The future research directions need to address the following issues: (1) to reduce the computing time because of the many iterative FE simulations at each backward increment; (2) to establish an algorithm for determining boundary conditions for each backward increment.

2.9.2 Minimisation of Variation in Effective Strain Rate

The purpose of this method is to make the deformation of the workpiece as uniform as possible by using reverse simulation. Han et al. [62] developed an optimisation technique for the design of preforms for H-shaped plane strain and axisymmetric forging operations. The technique uses the minimisation of the variation in effective strain rate within the workpiece to determine the sequence of contact nodes to be released from the die in the reverse simulation. The FE node with the smallest velocity is regarded as having the least influence on the deformation and is detached from the die during each reverse increment. Another similar study was given by Srinivasan et al. [63] in an application of axisymmetric components. These simulation results showed that complete die filling was obtained and the variation in accumulation plastic strain was reduced by using the proposed technique. Since the backwards tracing scheme and the

design sensitivity analysis which illustrates the priority of releasing a node were involved in the technique, reducing computing time and increasing numerical accuracy should be handled carefully.

2.9.3 Fuzzy Logic Making Decision

Fuzzy logic is a technique for representing and manipulating uncertain information. The early work in fuzzy logic was contributed by Lotfi A. Zadeh in the mid-1960s [35, 64]. In traditional logic, a fact, such as "the train will be on time", must be true or false. Much information however involves some degree of uncertainty, such as "young", "tall" and "famous" for describing a person. Fuzzy logic measures the degree of correctness of a statement in relation to the uncertain information provided by membership functions. The functions are based on parameters which represent uncertain information. For instance, a simple membership function based on the height h of the person in centimeter to measure the degree of tallness could be as follows:

$$Degree\ of\ Tallness(h) = \begin{cases} 0.0 & : \quad h < 120 \\ \frac{(h-120)}{60} & : \quad 120 \leq h \leq 180 \\ 1.0 & : \quad h > 180 \end{cases}$$

Other information, such as the age, may also be taken into account to construct another membership function for measuring the degree of youth. An appropriate rule, i.e. a fuzzy rule, constructed by these two membership functions is then used to evaluate input data (height and age in this example) for measuring the degree of correctness for a statement such as "the person is tall for his age" in fuzzy logic. Normally the membership functions and the fuzzy rule are defined by knowledge and observations from experimental work.

Biglari et al. [65] used fuzzy logic to predict preforms for an axisymmetric disk forging. The study was also based on the reverse simulation with the backwards tracing scheme. Three geometrical parameters and effective strain deviation of the workpiece were used to construct membership functions, and computer simulations were carried out to define fuzzy rules. The rules were then used to measure the node releasing priority to specify new boundary conditions for each backward increment in the reverse simulation. The study predicted an intermediate shape which ensured complete die filling and decreased the deviation of the plastic strain by 20 % compared with a previous design.

Fuzzy logic provides an alternative technique different to the traditional true or false logic to measure the degree of the correctness for a statement. To apply the technique to preform deign requires a method for representing the workpiece geometry by parameters to describe a complex forging shape. Furthermore, the definition of appropriate fuzzy rules based on reliable knowledge is also important. Solving these problems will be the direction of future studies in applying the fuzzy logic technique to reverse simulation.

2.9.4 Inversion and Modification of Contact Time Sequence

During the forward forging simulation, the boundary condition is identified by when a particular area of the die makes contact with the workpiece. The record of the contact time could be modified for a reverse simulation in order to improve or achieve the preform design. Zhao et al. [66, 67] developed guidelines based on the material flow characteristics and the state of die filling for modifying the boundary condition sequence from a forward simulation of a trial preform. The modified sequence was then used to determine the boundary conditions for the reverse simulation. A trial preform design is needed in this procedure, and it is

also necessary to develop a mathematical approach for modifying the boundary condition sequence.

2.9.5 Reducing Geometric Complexity

One of the objectives of preform design is to achieve the geometric requirement, i.e. to avoid the under-filling problem at the end of the forging process. The geometric complexity factor of the workpiece, which defines a relationship between the initial billet and the desired shape, increases after the deformation in most practical forging processes. This factor must be reduced or minimised during reverse simulations. Zhao et al. [68] used the FE simulation with the backwards tracing scheme to predict a preform for an axisymmetric forging. A shape factor is used to define the complexity of the workpiece in which the simplest shape is a cylinder, and the value of the factor increases during the forward forging process. The zero-one programming technique was employed to search for the lowest complexity resulting from the optimisation of contact conditions in which one of the FE nodes was released during the reverse simulation. The study showed that the preforms designed for an axisymmetric component achieved complete die filling, and positive benefits in forging load and deformation uniformity were realised. This design approach has potential in reverse simulation since a technique to deal with new boundary conditions has been proposed. However, reducing computing time is again of great concern for practical applications.

2.9.6 UBET Based Reverse Simulation

In 1974, McDermott and Bramley [69] proposed the upper bound elemental technique (UBET). The technique obtains upper bound solutions by dividing a work-

piece into elementary simple regions for plane strain and axisymmetric forging problems. This technique provides a fast simulation to predict an approximate metal flow and required forging loads. A reverse simulation with UBET was also proposed by Bramley [16] to predict preforms for axisymmetric forging problems. The preform was obtained by reversing the direction of the velocity field which gave a minimum value for the overall rate of energy dissipation and was optimised by the current boundary contact conditions. However, in the study, no attempt was made to control the boundary conditions which are important to the solution of the reverse simulation. To define the boundary conditions for the UBET reverse simulation, Kim et al. [70] used prescribed rules for controlling contact regions. The simulation also used the backwards tracing scheme to trace the loading path at each reverse increment. The predicted preforms, which were verified by FE simulations and experiments with lead, illustrated the viability of the design procedure for a connecting rod forging.

The UBET-based reverse simulation can be used to predict preforms in an early design stage due to its fast and approximate characteristics. The predicted preforms can then be evaluated by an FE forward simulation to examine the validity of the design. The FE result is analysed to assign new boundary conditions for a new trial UBET-based reverse simulation. The procedure is repeated until the required shape is achieved. This design approach was applied to determine an optimum billet for an axisymmetric gear blank in which the minimisation of flash was the objective of the design [71].

By using the upper bound method with the finite element technique, Pitt-Francis et al. [87] used a simple comparison factor to evaluate the complexity of deformed shapes resulting from different boundary conditions by releasing a contact FE node in each test through all contact nodes. The simplest shape, i.e. the shape with the smallest complexity factor, was then chosen as the deformed shape for

the next increment in the reverse simulation. The new FE formulation distributes the velocity field at nodes, and is different from UBET which considers the velocity field on the element boundaries. This FE-based reverse simulation approach may require high computing times since the number of the contact nodes is large.

The UBET-based reverse simulation is able to predict an approximate and rapid preform design. The predicted preform may be examined by the FE simulation and be modified according to the results. An alternative design method using the upper bound method with the FE formulation is able to deal with complex geometry and may possibly reduce computing times. However, as with other reverse simulation design approaches described in previous sections, a mechanism to control the boundary conditions is the main issue on future studies.

2.9.7 Summary of Reverse Simulation Techniques

In a forward simulation or a real forging process, the deformation occurs after dies contact the workpiece and the dies move forward. The boundary conditions for the simulation are determined according to the geometry of the workpiece and the position of the dies. This situation however does not exist during reverse simulations. Therefore, one of the main research objectives of reverse simulation is to establish an algorithm for defining a target shape aiming for a simple geometry. The algorithm should be able to specify when and where the workpiece will be detached from the die, and thus suitable boundary conditions can be controlled in the reverse simulation. The algorithm could be a trial and error or an optimisation approach but it may require high computing times. Thus, to develop a preform design system using reverse simulation, the following requirements should be fulfilled:

1. Reverse modelling of forging process — One of the methods is to use an iterative optimisation to modify the reversed shape which results in the forward deformation is the same as or very close to the current deformation.

2. Capability for dealing with complex geometry — This capability enables the reverse simulation for general use.

3. Determination of target shape — The target shape controls the deformation in terms of design requirements, e.g. a simple shape of the preform or billet.

4. Control of boundary conditions — A strategy defines new boundary conditions for each reverse increment during the simulation.

2.10 Summary of Literature Review

Conventional industrial preform design is based on the existing design experience and relevant knowledge in the forging industry. This approach might be time consuming or fail when a new type of design is encountered for which there is no existing design knowledge. A knowledge-based system capturing the existing industrial experience is able to provide an aid for designers to judge or design preforms and processing conditions. These systems are especially useful for forged products which can be classified by geometric features. In order to validate an existing empirical design, FE forward simulation can be employed to predict the deformation process from the initial billet to the final shape. The simulation results also provide guidelines for the modification of the current design if the demanding quality is not satisfied. This is obviously an expensive trial and error procedure but better than a practical one. To search for the optimal preform subject to a particular requirement, such as the uniform strain distribution, an optimisation technique can be combined with FE simulation to

establish a systematic design system. The technique modifies the trial design in terms of iterations. Furthermore, techniques from biological science, such as genetic algorithms and neural networks, have also been combined with the FE simulation and applied to preform design. There is no doubt about the validity using these forward simulation approaches to design acceptable or the optimal tools or preforms in terms of complete die filling or quality. Nevertheless, high computing costs or the lack of a technique for presenting geometric features are the main obstacles in applying such FE based design approaches in the forging industry.

Preform design may use a method of reverse simulation which starts from the final product shape and ends with an initial billet or a preform. This provides an alternative design route without the need for a trial initial workpiece and preform geometry. In other words, the reverse simulation can reduce the cost of forging design if an appropriate modelling technique can be established. Simulation has been developed using different techniques such as the backwards tracing scheme with FE simulation, UBET, and fuzzy logic for controlling the boundary conditions. These techniques, although still needing improvement, provide potential approaches to establishing an automatic preform design system.

According to the literature review in previous sections, many design methods are based on FE simulations, which may be time-consuming but are able to provide detailed information about the deformation of the forging process. FE simulation is also capable of dealing with complex geometry of the workpiece which is the main drawback of the UBET used for the rapid simulation in forward mode as well as in reverse. A combination of these two techniques could therefore be adopted to improve current upper bound based methods of preform design.

2.11 Project Objectives

As noted in previous sections, one of the objectives of forging preform design is to fill the die cavity efficiently and completely with a minimum amount of dissipation energy (or forging load) in a manner which will prevent the occurrence of forging defect such as folds or overlaps. This is an essential requirement for preform design. Furthermore, reverse simulation predicts preforms without a trial initial geometry of the workpiece and could be an efficient design approach. The aim of this thesis is therefore focused on establishing a preform design approach using the reverse simulation subject to the complete filling of the die cavity.

The objectives of this thesis are as follows:

1. To model forging process in reverse for complex geometry — The upper bound theorem with the FE formulation is used for this purpose. This work is continued by extending the development of TEUBA.

2. To establish an algorithm for determining a target shape — A concept of material distribution to control the target shape is proposed to fulfil the requirement of complete die filling.

3. To establish a technique for defining the boundary conditions — The technique identifies a possible region to be released according to the target shape and the deformed workpiece.

4. To evaluate the proposed preform design by performing one-stage and multi-stage forging processes.

Chapter 3

Upper Bound Theorem and Forging Modelling

3.1 Introduction

Limit analysis theorems, which deal with the determination of the loading capacity of structures made of perfectly plastic materials, were introduced in 1951 by Drucker, Greenberg and Prager. These theorems were deduced from part of work principles published in 1950 by Hill [72]. It however appears that the earliest reference to the theorems of limit analysis was probably due to Gvozdev whose paper in 1936 was translated in 1960 by Haythornthwaite [73, 74, 75].

Limit analysis involves two principles which lead to both the lower bound and upper bound methods. The lower bound method predicts a load which is less than or equal to the exact limit load needed to enable the plastic flow of metals. The solution of the method must satisfy equilibrium equations, a suitable yield criterion, and static boundary conditions. On the other hand, the upper

bound method provides a load which is at least equal to or greater than the exact limit load, and obtains a kinematically admissible velocity field which must satisfy kinematic boundary conditions. Since the upper bound method provides an estimate for the forging load which ensures that the metal forming operation can be performed, it is therefore useful for analysing metal forming process.

In 1960, Kudo [76, 77] proposed a method using the concept of unit triangular and rectangular deforming regions to obtain upper bound solutions for plane strain forging, axisymmetric forging and extrusion problems. The technique was later improved by Kobayashi [78, 79] who incorporated curved boundaries for the regions to improve the solution of axisymmetric problems. In 1974, Kudo's ideas were then elucidated in practical applications by McDermott and Bramley [69] who demonstrated an upper bound elemental technique (UBET) by which quite complicated forging shapes could be divided into elements of simple triangular, rectangular or curved regions. Each of the elements is then treated by the rule of the upper bound theorem, and the whole solution is assembled within a computer program to establish the forging load. With the improvement of the precision by an optimising procedure, UBET has been successfully used to analyse the axisymmetric [80] and plane strain forging problems [81]. Flow prediction and the effect of flash dimensions on forging have been studied by using UBET to analyse and optimise forging parameters in axisymmetric cases [82]. The different types of the elements have also been formulated for solving three-dimensional metal forming problems [38, 83, 84, 85] to predict the forging load and the velocity field. A brief list of the types of the UBET elements is reviewed by Lee et al. [85].

The UBET assumes that the flow pattern is a single velocity field in each of the elements. This assumption makes the technique difficult to obtain more realistic flow patterns and deal with more complex forging shapes, although the

incompressibility can easily be achieved by an appropriate formulation for different types of the element. A finite element formulation could be a method to overcome this difficulty.

Recently, Liu et al. [86] proposed a numerical procedure based on the upper bound method with the finite element formulation to investigate the limit analysis of cylindrical shells under internal pressure, and the numerical results showed the applicability of the proposed procedure. Pitt-Francis et al. [87] and Hwan [88, 89] formulated the upper bound theorem using linear tetrahedral elements in three dimension and linear quadrilateral elements in plane strain respectively with finite element procedures to solve forging problems. These formulations distribute a kinematically admissible velocity field at the nodes of the finite elements to form a limit analysis problem. According to the obtained velocity field, a new deforming shape can be updated for a small time step. A subsequent limit problem is then solved for the new deformed shape. The updating process is repeated to form a sequence leading to the solution of a forging problem and is referred to as sequential limit analysis which is the basis of the modelling technique developed in this thesis.

3.2 Forging Modelling

Forging modelling, in general, predicts (1) metal flows which provide information including the strain and the stress distribution in the workpiece; and (2) the force and energy required to perform a forging operation [90]. These results can then be used to adjust variables of forging system to obtain the desired shape and properties, or to examine whether defects exist on or inside the product after the operation.

To model a complicated forging process, it may be necessary to make some simplifying assumptions that have less effect on required information about the evaluation of the design; or may ignore some variables which are not significant factors in analysing the forging process from experimental observations or experiences. The simplification of the forging process should consider (1) forging conditions including strain, strain rate, friction and temperature; and (2) material behaviour under the assumed process conditions.

3.2.1 Assumption of Forging Conditions

The forging operation is designed to achieve the desired process conditions which control the material behaviour of workpiece. The process variables include the geometry of both workpiece and tools, the speed of tools, the lubrication conditions between the workpiece and tools, and the temperature of both workpiece and tools. In order to make appropriate assumptions for modelling a forging process, it is important to understand which variables are critical.

Tool Velocity — Strain Rate, Operating Time and Heat Transfer

A forging operation under different strain rate conditions, which may result from using different forging machines, may require different forging loads and energies. For example, the maximum load in a drop hammer process, which is a high strain rate process, may be lower than that of a hydraulic press. The phenomenon is different from the result of material test which shows that a higher load is required in a higher strain rate condition. The reason is different heat transfer conditions in these two cases. In the hydraulic press operation the flash area cools more obviously, however the temperature remains nearly the same as the initial temperature in the drop hammer process because the operation time is short [1, 90]. Moreover, in the drop hammer forging process, the contact time is

short. Thus, after the deformed workpiece has been removed from the dies, the tool temperature can decrease for a considerable period of time until the next operation is performed.

Temperature — Operating Temperature and Generated Heat

The initial temperatures of the workpiece and tools may be determined according to the design. However, plastic deformation generates heat which remains in the deformed material or flows into dies, atmosphere or un-deformed material. The temperature may increase several hundred degrees in some forming operations, such as drawing and extrusion, performed at high speed. This generated temperature modifies the constitutive relation for the workpiece and is thus a significant factor in determining the maximum deformation speed that can be used to produce good products without excessive tool damage. It also affects the lubrication condition and hence tool life [90]. The main factors influencing the magnitude and distribution of the temperature include the initial material and tool temperatures, the heat generation due to the plastic deformation, the friction and the heat transfer between the workpiece, tools and environment.

Lubrication — Friction Effects

In forging process, the applied force is transmitted from the dies to the deforming workpiece. Therefore, the frictional conditions at the die-workpiece interface influence the formation of surface stresses acting on the interface and thus metal flow. Moreover, friction generates heat which changes temperature distributions, and hence affects the material behaviour and forging load and energy. To control the frictional conditions requires use of appropriate lubricants for particular applications. The lubricant may be applied simply to prevent sticking and reduce the sliding friction and thus increase the life of dies.

It is apparent from above that appropriate assumptions about the forging con-

ditions are important and should be taken into account according to the characteristics of the forging process. In this thesis, the assumptions are as follows:

1. Isothermal forging process — The temperature effects are not taken into account. This isothermal assumption may be applied to a forging operation in which the temperature is well controlled or its effects are small.

2. Constant velocity of tools — This assumption can be applied to some forging processes, such as forging in hydraulic press.

3. Rigid tools — The deformation of tools compared with that of workpiece is negligible. This is reasonable for most forging operations, especially in cases where further machining of the forged product is needed.

4. Friction exists at tool-workpiece interface.

3.2.2 Flow Formulation for Modelling Material Behaviour

Large plastic deformation problems involving metals, such as forging processes, can generally be solved by two kinds of formulation, known as solid formulation and flow formulation [90]. Solid formulation [91, 92] considers the behaviour of the material as a classical elastic-plastic solid. The formulation can predict a complete stress-strain history which is thus useful for the analysis of residual stress. However, this formulation has the limitation to the magnitude of strain and rotation increments, which may make the method rather expensive in computation. In the analysis of hot metal forming process, elastic strain of a deformed metal may be neglected since it is usually very small as compared to plastic strain. By neglecting elastic strains a deforming metal can be treated as a fluid to simplify the formulation. It is then possible to relate the stress σ to the strain rate $\dot{\epsilon}$ in

the following general form

$$\sigma = \mathbf{C}\dot{\epsilon} \tag{3.1}$$

where the quantity $\mathbf{C}$ may be dependent on the effective strain $\bar{\epsilon}$, the strain rate $\dot{\epsilon}$ and temperature T, i.e.

$$\mathbf{C} = \mathbf{C}(\bar{\epsilon}, \dot{\epsilon}, T) \tag{3.2}$$

The formulation based on this simplification is known as a flow formulation [93]. It can provide useful information with lower computing costs than a solid formulation. However, the stress distribution of flow formulation is only determined by current strain rate distribution. The solution is not affected by the previous stress state. Therefore, it cannot predict the complete stress-strain history for the analysis of residual stress. Nevertheless, some improvements have been made to flow formulation to deal with residual stress [94] whereas the solid formulation has also been improved so that it can deal with large strain increments [92].

Defining Plastic Deformation

Plastic deformation can be defined as follows:

1. Constitutive equations — These equations describe the relationship between stress and strain during plastic deformation.

2. Yield criterion — The criterion, a function of the stress tensor, predicts a stress state required for plastic deformation.

3. Equilibrium equations — The upper bound theorem used in this thesis disregards these requirements.

4. Compatibility equations — These equations, which are relationships between displacements and strains, maintain the continuum requirement without fracture or crack formation.

5. Boundary conditions — Surface tractions, displacements (or velocities) and temperature distribution are the main boundary conditions in the forging process. In this thesis, isothermal conditions are assumed and the equilibrium equations are neglected, and thus the velocity is the only type of the boundary conditions.

In flow formulation, the relationship between stress and strain (or strain rate) is represented by a flow rule which is associated with a yield criterion. These definitions and the virtual work principle are used to construct the weak form formulation, which describes the system in terms of energy, for modelling the forging process. One of the above definitions may be disregarded to simplify the complexity of the modelling. This thesis employs the upper bound theorem, which neglects equilibrium equations, with flow formulation to model the forging process.

3.3 Flow Rule Associated with von Mises Yield Criterion

To analyse the forging process, it is necessary to develop a yield criterion and a flow rule. The yield criterion in terms of mathematical relationships predicts when the plastic deformation, i.e. yield, occurs. The flow rule, a relationship of stress and strain during plastic deformation, describes how and where the material will flow.

von Mises Yield Criterion

Two criteria that are most generally used to predict the yielding are (1) Tresca (maximum shear stress) criterion, and (2) von Mises criterion. In flow formu-

lation, it is assumed that the normality principle is fulfilled. Some singularity points may be encountered by using Tresca criterion, and hence require special treatment in any generalised formulation. On the other hand, von Mises criterion can be easily applied to the analysis and predicts yielding for most engineering metals with sufficient accuracy [75]. Therefore, the von Mises criterion is commonly used in flow formulation for modelling the forging process.

von Mises (1913) proposed a yield criterion, which states that plastic deformation begins when shear strain energy reaches a critical value k dependent upon the characteristics of the material. From the experimental observations, the hydrostatic pressure does not cause yield in a continuous solid [9]. Therefore, von Mises yield criterion in terms of the deviatoric stress s_{ij} is written in the form

$$J_2 - k^2 = 0 \tag{3.3}$$

where J_2 is the second invariant of the deviatoric stress, and

$$s_{ij} = \sigma_{ij} - \frac{1}{3}\delta_{ij}\sigma_{kk} \tag{3.4}$$

$$J_2 = \frac{1}{2}s_{ij}s_{ij} \tag{3.5}$$

where δ_{ij} is the Kronecker delta. By comparison of behaviour in simple compression and tension tests it may be shown that

$$k^2 = \frac{Y^2}{3} \tag{3.6}$$

where Y is the yield stress in simple compression test.

Flow Rule

The flow rule is a kinematic assumption postulated for plastic deformation or plastic flow. It gives the ratio or relative magnitude of the components of the

plastic strain increment tensor $d\epsilon_{ij}$. An alternative expression of plastic strain increment is in terms of strain rates $\dot{\epsilon}_{ij}$. The increment $\dot{\epsilon}_{ij}$ may be represented by a vector with nine components in a strain space. The flow rule therefore also defines the direction of the plastic strain increment vector $\dot{\epsilon}_{ij}$ in the strain space. The plastic flow equations representing increments of strain related to stress can be written in the form

$$\dot{\epsilon}_{ij} = \lambda \frac{\partial G}{\partial \sigma_{ij}} \tag{3.7}$$

where

$$G = G(\sigma_{ij}, \epsilon_{ij}) \tag{3.8}$$

is a plastic potential function and λ is a positive scalar factor of proportionality, which is nonzero only when plastic deformations occur. The simplest case of defining the plastic potential function assumes that the plastic potential and the yield function F coincide, i.e. $G = F$. Thus,

$$\dot{\epsilon}_{ij} = \lambda \frac{\partial F}{\partial \sigma_{ij}} \tag{3.9}$$

Equation (3.9) is called the associated flow rule because the plastic flow is connected or associated with the yield criterion.

Flow Rule Associated with von Mises Yield Criterion

For an isotropic material, the yield function F in equation (3.9) may be defined by von Mises yield criterion.

$$F = J_2 - k^2 \tag{3.10}$$

Evaluating the derivative in equation (3.9) gives

$$\frac{\partial F}{\partial \sigma_{ij}} = \frac{\partial}{\partial \sigma_{ij}} J_2 = \frac{\partial}{\sigma_{ij}} (\frac{1}{2} s_{ij} s_{ij}) = s_{ij} \tag{3.11}$$

Using equation (3.9) and (3.11), the flow rule may be written as

$$\dot{\epsilon}_{ij} = \lambda s_{ij} \tag{3.12}$$

For a von Mises material model, the effective stress $\overline{\sigma}$ can be shown to be

$$\overline{\sigma} = \sqrt{3J_2} = \sqrt{\frac{3}{2} s_{ij} s_{ij}} \tag{3.13}$$

and the effective strain rate $\dot{\overline{\epsilon}}$ may be expressed as

$$\dot{\overline{\epsilon}} = \sqrt{\frac{2}{3} \dot{\epsilon}_{ij} \dot{\epsilon}_{ij}} \tag{3.14}$$

Using equation (3.12), (3.13) and (3.14) obtains the scalar λ

$$\lambda = \frac{3}{2} \frac{\dot{\overline{\epsilon}}}{\overline{\sigma}} \tag{3.15}$$

Therefore,

$$\dot{\epsilon}_{ij} = \frac{3}{2} \frac{\dot{\overline{\epsilon}}}{\overline{\sigma}} s_{ij} \tag{3.16}$$

or

$$s_{ij} = \frac{2}{3} \frac{\overline{\sigma}}{\dot{\overline{\epsilon}}} \dot{\epsilon}_{ij} \tag{3.17}$$

which is the Levy-Mises equation.

For a rigid-perfectly plastic material, the effective stress is expressed in the form

$$\overline{\sigma} = Y = constant \tag{3.18}$$

The flow rule associated with von Mises yield criterion defines the relationship between the deviatoric stress and strain rate for the rigid-perfectly plastic material

as

$$s_{ij} = \frac{2}{3}\frac{Y}{\dot{\bar{\epsilon}}}\dot{\epsilon}_{ij} \tag{3.19}$$

3.4 Upper Bound Theorem

3.4.1 Assumptions

Three basic relations, which are the equilibrium equations, the compatibility equations, and the constitutive equations, should be satisfied for a valid solution for any problem in mechanics of solids. The compatibility equations ensure that the displacements are single-valued and continuous, i.e. a crack formation does not exist during the deformation. The constitutive equations, i.e. the relationship between stress and strain, describe the behaviour of the material after applying a force. Once the constitutive relations for a material are established, i.e. when the equilibrium equations and the compatibility equations are linked, the general formulation for the solution of a solid mechanics problem can be completed. However, as has been noted in the previous discussions, a complete forging process analysis is generally quite complicated. Therefore, the development of more efficient methods that can be used to obtain information, such as the collapse load of a structural problem or the load of a metal forming problem, is of great value to engineers despite the fact that the information obtained is only a part of the total solution. Limit analysis theorems, the upper bound and lower bound theorems, were developed to provide an estimate of the collapse load of a structure in an approximate and efficient manner. In general, only the equilibrium equations and yield criterion need to be satisfied for a lower bound solution, and only compatibility equations and the flow rule associated with a yield criterion need to be fulfilled for an upper bound solution [95].

In this thesis, the upper bound theorem with the following assumptions for material behaviour is used to model the forging process.

1. The material is isotropic, homogeneous and incompressible.
2. The material is rigid-perfectly plastic without strain hardening or softening.
3. The material has a convex yield surface and obeys normality.
4. The deformation is small and negligible at the limit load.

3.4.2 Theorem

The upper bound theorem considers a kinematically admissible velocity (or displacement) field which satisfies;

1. the velocity boundary conditions,
2. the compatibility conditions, and
3. a material flow rule

to describe the deformation of the material and to estimate the collapse load which induces the plastic deformation. No attempt is made to satisfy the stress (or force) equilibrium conditions at any point in the material [95, 96].

Consider that a proposed displacement field u_{ij}^U corresponding to strain field ϵ_{ij}^U fulfils a prescribed displacement boundary conditions in a body of volume V with the boundary surface S. According to the principle of virtual work,

$$\int_V \sigma_{ij}^A d\epsilon_{ij}^U dV = \int_S T_i^A du_i^U dS \tag{3.20}$$

where σ_{ij}^A and T_i^A are the actual equilibrium stress and the actual traction field respectively during collapse, i.e. plastic deformation, and the compatibility equations must be satisfied as well. Furthermore, the material is rigid-perfectly plastic, obeys convexity and normality, and thus the principle of maximum plastic work holds that

$$\int_V (\sigma_{ij}^U - \sigma_{ij}^A) d\epsilon_{ij}^U dV \geq 0 \tag{3.21}$$

$$\int_V \sigma_{ij}^U d\epsilon_{ij}^U dV \geq \int_V \sigma_{ij}^A d\epsilon_{ij}^U dV \tag{3.22}$$

where σ_{ij}^U is produced stress field after increment strain field $d\epsilon_{ij}^U$ was applied. It is clear when equation (3.20) and (3.22) are combined that

$$\int_V \sigma_{ij}^U d\epsilon_{ij}^U dV \geq \int_S T_i^A du_i^U dS \tag{3.23}$$

It is important to note that the equilibrium equations are not satisfied generally, in such a case $\sigma_{ij,j}^U \neq 0$, although the proposed ϵ_{ij}^U is a compatible strain field. The stress distribution is not necessary in equilibrium, and is only defined in the deforming region because the material is assumed as rigid-perfectly plastic. The upper bound theorem is defined by setting the external plastic work increment equal to the internal work increment, i.e. energy equilibrium,

$$\int_S T_i^U du_i^U dS = \int_V \sigma_{ij}^U d\epsilon_{ij}^U dV \tag{3.24}$$

From equation (3.23) and (3.24), the following equation can be obtained.

$$\int_S T_i^U du_i^U dS \geq \int_S T_i^A du_i^U dS \tag{3.25}$$

Equation (3.24) and (3.25) written in the time rate form are:

$$\int_S T_i^U \dot{u}_i^U dS = \int_V \sigma_{ij}^U \dot{\epsilon}_{ij}^U dV \tag{3.26}$$

$$\int_S T_i^U \dot{u}_i^U dS \geq \int_S T_i^A \dot{u}_i^U dS \tag{3.27}$$

Therefore, this concludes that [75]

> *"The rate of work done by the actual surface traction with prescribed velocities is less than or equal to the rate of work done by the surface traction corresponding to any other kinematically admissible velocity field."*

The upper bound theorem predicts a load which is at least equal to or greater than the exact actual load needed to deform the material plastically. Therefore, the minimisation of the work done by the external traction to satisfy the prescribed boundary conditions will provide the best upper bound analysis. In terms of the upper bound theorem, the problem of modelling plastic deformation is described by minimising the plastic deformation power equation (3.26) as follows (Note that the superscript U is ignored for convenience.):

$$\begin{aligned} &\textit{minimise} && \int_V \sigma_{ij}\dot{\epsilon}_{ij} dV \\ &\textit{subject to} && \textit{incompressibility} \\ & && \textit{velocity boundary conditions} \end{aligned} \tag{3.28}$$

3.4.3 Upper Bound Theorem with Flow Rule

The workpiece is assumed to be a rigid-perfectly plastic material, i.e. there is no elasticity and no work hardening or softening. This is a reasonable approximation for most formable materials at high temperatures [97]. In terms of the deviatoric

stress s_{ij}, the plastic deformation power in equation (3.28) is written as

$$\int_V (s_{ij} + \delta_{ij}\sigma_{kk})\dot{\epsilon}_{ij} dV \tag{3.29}$$

and due to the incompressibility $\dot{\epsilon}_{ii} = 0$,

$$\delta_{ij}\sigma_{kk}\dot{\epsilon}_{ij} = \sigma_{kk}\dot{\epsilon}_{ii} = 0 \tag{3.30}$$

Using the flow rule associated with von Mises yield criterion, i.e. equation (3.19),

$$s_{ij} = \frac{2}{3}\frac{Y}{\dot{\bar{\epsilon}}}\dot{\epsilon}_{ij}$$

the plastic deformation power for rigid-perfectly plastic material becomes

$$\int_V \frac{2}{3}\frac{Y}{\dot{\bar{\epsilon}}}\dot{\epsilon}_{ij}\dot{\epsilon}_{ij} dV \tag{3.31}$$

Substituting the effective strain rate, equation (3.14),

$$\dot{\bar{\epsilon}} = \sqrt{\frac{2}{3}\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}$$

into the plastic deformation power gives

$$\int_V \sqrt{\frac{2}{3}} Y \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}} dV \tag{3.32}$$

Let $Y^U = \sqrt{\frac{2}{3}}Y$, the equation becomes

$$\int_V Y^U \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}} dV \tag{3.33}$$

Therefore, the problem of modelling plastic deformation (3.28) is written as follows:

$$\begin{aligned} &\textit{minimise} && \int_V Y^U \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}\, dV \\ &\textit{subject to} && \textit{incompressibility} \\ & && \textit{velocity boundary conditions} \end{aligned} \tag{3.34}$$

3.5 Incompressibility

A kinematically admissible velocity field that satisfies the compatibility equations should also meet the incompressibility requirement, i.e. the volume strain rate equals zero, $\dot{\epsilon}_{kk} = 0$. This requirement has been added into the upper bound problem (3.34) as a constraint which ensures that the minimum energy dissipation is only dependent on internal shear deformation if friction effects are not taken into account. Two methods, the Lagrange multiplier and the penalty method, are most frequently used to deal with the incompressibility constraint in minimisation problems.

3.5.1 Lagrange Multiplier Method

Consider a function $\int_\Omega f(x)dx$ that is to be minimised subject to a constraint condition, $g(x) = 0$, which is differential in domain Ω. The Lagrange multiplier method defines a function M to be $M(x) = \int_\Omega (f(x) + \eta g(x))dx$, where η is a new variable called the Lagrange multiplier. The variables, x and η, are then solved by equations, $\frac{\partial M}{\partial x} = 0$ and $\frac{\partial M}{\partial \eta} = 0$, to find the minimum value. It can be seen that the total number of unknowns has increased when using the Lagrange

multiplier method [93, 98].

3.5.2 Penalty Method

Consider the same minimisation problem $\int_\Omega f(x)dx$ and the constraint $g(x) = 0$. The penalty method [93, 99] defines a function $I(x) = \int_\Omega (f(x) + \alpha[g(x)]^2)dx$, where α is a large positive real number. In order to minimise the function $I(x)$, it is necessary to ensure $\int_\Omega [g(x)]^2 dx$ is close to zero as α is a large positive number. This can only be true if $g(x)$ is close to zero everywhere in the domain Ω, this almost satisfies the constraint $g(x) = 0$. However, if α is small, the constraint can not be satisfied. On the other hand, if α is too large, numerical errors may increase. Therefore, it is necessary to use an appropriate α value. The value is determined by experience.

Due to the drawback of increasing unknown variables by using the Lagrange multiplier method, the penalty method is employed to deal with the incompressibility requirement in this thesis. The upper bound problem (3.34) using the penalty method is then transferred into

$$\begin{aligned} &\textit{minimise} \quad \int_V Y^U \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}\, dV + \int_V \alpha(\dot{\epsilon}_{kk})^2\, dV \\ &\textit{subject to} \quad \textit{velocity boundary conditions} \end{aligned} \tag{3.35}$$

where the penalty constant α may be interpreted as a constant to the bulk modulus [90] because the mean stress σ_m is related to the volumetric strain rate $\dot{\epsilon_v}$ by $\sigma_m \cong \alpha\dot{\epsilon_v}$.

3.6 Frictional Boundary Conditions

Friction between the workpiece and dies greatly influences the metal flow, the product surface and internal defects, pressure applied on the dies, and the load and energy requirements. It is therefore necessary to take the friction effects into consideration in the analysis model in terms of a quantitative friction factor. Work done by friction may then be expressed by a rate form

$$\int_{S_f} \tau_i^f \dot{u}_i^r dS \tag{3.36}$$

where $\dot{u}_i^r$ is the relative velocity vector used to reflect that the friction stress τ_i^f should be in the opposite direction to the relative motion between the workpiece and dies.

The most common form of friction law is known as Coulomb's law, $F_f = \mu F_n$ where F_f is the friction force and F_n is the normal force. The form may be expressed in a stress field by $f_s = \mu\sigma_n$ in which f_s and σ_n are friction stress and the normal stress respectively. μ is called the friction coefficient. This law is usually found applicable at low contact pressures relative to material strength before there is a great deal of deformation. However, in the forging process contact pressures are generally high. Coulomb's law drastically overestimates friction [96]. Due to the yielding process, the tangential stress required to induce slip which causes deformations is dependent on the shear strength of the material. Moreover, the friction at the interface can not exceed the shear strength and velocities are the variables used to model plastic deformation in the upper bound theorem. A model, Tresca friction model [90], in terms of the yield shear stress k and the relative velocity vector $\dot{u}_i^r$ is used to reflect the friction stress τ_i^f in the

form

$$\tau_i^f = mk\frac{\dot{u}_i^r}{\|\dot{u}_i^r\|} \tag{3.37}$$

where m is a friction factor between 0 and 1 to represent a fraction of the yield shear stress, and k equals $\frac{Y}{\sqrt{3}}$ since Tresca yield stress is involved. In order to obtain the friction factor m, the ring compression test described in section 7.3.4 is widely used.

Considering friction effects, the external work in equation (3.26) should also include the work done by the friction stress. The frictional term is therefore added into equation (3.26) as

$$\int_S T_i^U \dot{u}_i^U dS = \int_V \sigma_{ij}^U \dot{\epsilon}_{ij}^U dV + \int_{S_f} \tau_i^f \dot{u}_i^r dS \tag{3.38}$$

Finally, an upper bound problem (3.35) involving friction effects can therefore be stated as

$$\begin{aligned} &\textit{minimise} && \int_V Y^U \sqrt{\dot{\epsilon}_{kl}\dot{\epsilon}_{kl}}dV + \int_V \alpha(\dot{\epsilon}_{kk})^2\, dV + \int_{S_f} \tau_i^f \dot{u}_i^r dS \\ &\textit{subject to} && \textit{velocity boundary conditions} \\ &&& \textit{sliding contact friction constraints} \end{aligned} \tag{3.39}$$

where sliding contact friction constraints enforce the relative velocity vector $\dot{u}_i^r$ in the tangential direction at the frictional interfaces.

3.7 Summary of Forging Modelling

A modelling method based on the upper bound theorem has been developed to solve the large deformation problem of the forging process. The method assumes

that the forging operation is an isothermal process with the conditions of constant die velocity and rigid dies. The friction effects at the interface between the workpiece and dies are also taken into account. Moreover, the workpiece is isotropic and homogeneous, and follows a rigid-perfectly plastic behaviour under an incompressibility constraint which enforces that the hydraulic pressure does not affect the plastic deformation. Finally, in terms of the minimisation of plastic work, a kinematically admissible velocity field which satisfies the velocity boundary conditions and the compatibility equations can be predicted. The geometry of the workpiece is updated by multiplying the predicted velocity field with a small time increment to provide a corresponding displacement. A subsequent upper bound problem is then solved for the new geometry. Repeating this procedure finally forms a sequence of limit analysis resulting in the solution of a forging deformation problem.

Since the equilibrium equations are released, stresses do not appear in this upper bound formulation. As a result, complicated stress updating is not needed in this upper bound analysis. The effect of material nonlinearity is incorporated in the analysis by using a flow rule associated with von Mises yield criterion. The analysis incrementally deals with deformation history. Using such an increment analysis of the forging process, it is also possible to solve strain hardening or softening material.

Like most flow formulations, the developed upper bound formulation cannot predict certain information, such as elastic strain and residual stress, which can only be obtained by an elasto-plastic incremental analysis. However, the upper bound formulation can provide most important information including forging load and deformation with reasonable computing cost.

Chapter 4

Finite Element Implementation

4.1 Introduction

The upper bound theory considers the kinematical velocity field in a continuous domain in which most forging problems are difficult or even impossible to be solved. The discretisation of the velocity field is therefore required. The finite element procedure is applied for this purpose.

The velocity field in the workpiece is distributed within each finite element in terms of the values at nodal points using interpolation functions, i.e. shape functions. The functions can be linear or non-linear and depend upon the type of element. For reduced computing time, three-noded triangular elements are used to model the forging problem in a discrete domain. The components of the velocity at each node become variables to define a forging problem with the boundary conditions.

By minimising the dissipation energy for plastic deformation, a set of non-linear

equations is obtained. Since the equations are solved by the Newton-Raphson method described in Chapter 5, the formulations of the first and second derivatives with respect to each velocity component are thus required. These formulations which are based on a finite element procedure are described in this chapter.

As mentioned in Chapter 1, Section 1.2 "Previous Work of TEUBA", Dr. Joe M. Pitt-Francis had implemented the finite element formulation of the upper bound theorem into TEUBA before this study started. This chapter therefore summarises the formulation referring to Pitt-Francis's paper [87], report [130], and the computer code of TEUBA. Since the intermediate formulations were not given in these references relating to the TEUBA code, this chapter makes a clear and complete presentation of the finite element formulation for the upper bound theorem.

4.2 Modelling in Discrete Domain

4.2.1 Discretisation of Velocity Field

The velocity field of the workpiece u_i is assumed to be a linear form for two-dimensional plane strain

$$\begin{aligned} u_i &= a_{i0} + a_{i1}x_1 + a_{i2}x_2 \\ &= a_{i0} + \sum_{j=1}^{2} a_{ij}x_j \end{aligned} \tag{4.1}$$

for each i $(1 \leq i \leq 2)$. By using the finite element method, the entire velocity

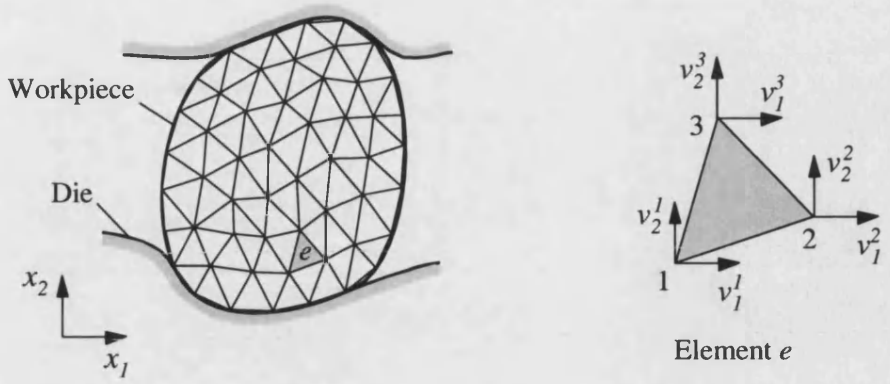


Figure 4.1: Three-noded triangle elements for the discretisation of the velocity field in the workpiece and nodal velocities in element e.

field of the mesh can be represented as

$$\dot{u}_i = \begin{cases} a_{i0}^e + \sum_{j=1}^{2} a_{ij}^e x_j & \text{in element } e \\ 0 & \text{elsewhere} \end{cases} \tag{4.2}$$

The velocities match at the boundaries of neighbouring elements, and are represented by a piecewise linear shape function $\phi^j(\mathbf{x})$ which has value 1 at the node j and is zero at other nodes. Thus,

$$\dot{u}_i = \sum_{j=1}^{N} \phi^j(\mathbf{x}) v_i^j \tag{4.3}$$

where v_i^j is the component of the velocity in the direction i at the j-th node, and N is the total number of nodes.

4.2.2 Discretisation of Relative Sliding Velocity at Contact Interface

The relative velocity vector $\dot{u}_i^r$ can be approximated in terms of nodal values v_i^r using shape functions, of two-noded line elements, which contact with the die

surface. Thus,

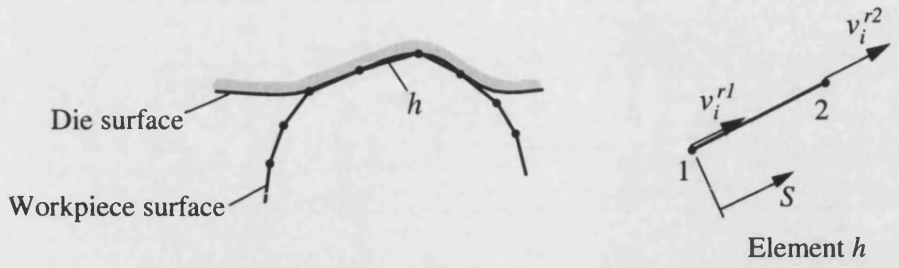


Figure 4.2: Two-noded line element for modelling friction effects and relative sliding velocities in element h.

$$\begin{aligned} \dot{u}_i^r &= \sum_{j=1}^{2} \psi^j v_i^{rj} \\ &= \sum_{j=1}^{2} \psi^j (v_i^j - v_i^{die}) \end{aligned} \tag{4.4}$$

where ψ^j is a linear shape function which has value 1 at the node j, and is zero at the other node. v_i^{rj} and v_i^j are the relative velocity and the nodal velocity at the node j, respectively, and v_i^{die} is the velocity of the die.

4.2.3 Enforcement of Sliding Friction Boundary Conditions

The sliding contact conditions take place at the interface between the workpiece and die surfaces. To enforce the contacting finite element nodes on the sliding surfaces, the constraint equations method [106] is employed. The procedure is implemented with a penalty method which is similar to that dealing with the incompressibility condition.

During the forging process, a contacting node j, Figure 4.3, is sliding on a die surface with the normal n_i^j. Therefore, the relative velocity vector v_i^{rj} in the

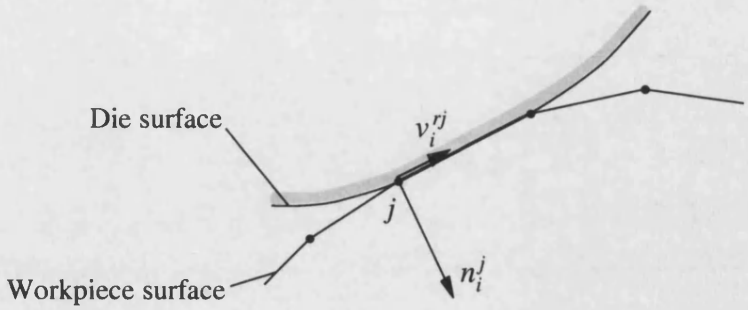


Figure 4.3: Node j in contact with a die surface and sliding on the surface with a normal direction n_i^j.

normal n_i^j direction of the die surface must be zero, i.e. $v_i^{rj} n_i^j = 0$. This provides constraint equations to impose the sliding contact condition. By using constraint equations $v_i^{rj} n_i^j = 0$ with a penalty factor β, the sliding contact friction constraints in equation (3.39) can be replaced by a penalty term

$$\Pi^S = \beta \sum_{j=1}^{H} (v_i^{rj} n_i^j)^2 \tag{4.5}$$

where H represents the number of the nodes in contact with die surfaces.

4.2.4 Upper Bound Theory with Finite Element Procedure

The upper bound theory is formulated in a discrete domain using three-noded triangle elements for the workpiece and two-noded line elements for modelling friction effects. The sliding contact friction constraints are also treated by a penalty method similar to the method dealing with the incompressibility. Therefore, the upper bound problem, i.e. equation (3.39), is transferred into a finite element domain as follows:

$$\textit{minimise} \quad \Pi^P + \Pi^I + \Pi^F + \Pi^S \tag{4.6}$$

subject to *velocity boundary conditions*

$$\Pi^P = \int_A Y^U \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}\, dA = Y^U \sum_{e=1}^{E} \int_{A_e} \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}\, dA$$

$$\Pi^I = \int_A \alpha(\dot{\epsilon}_{ii})^2\, dA = \sum_{e=1}^{E} \int_{A_e} \alpha(\dot{\epsilon}_{ii})^2\, dA$$

$$\Pi^F = \int_{S_f} \tau_i^f \dot{u}_i^r\, dS = \sum_{h=1}^{H} \int_{L_h} \tau_i^f \dot{u}_i^r\, dS$$

$$\Pi^S = \beta \sum_{j=1}^{H} (v_i^{rj} n_i^j)^2$$

where E is the number of elements and A_e is the part of the domain A occupied by the element e, and H is the total number of the elements in contact with die surfaces and L_h is the length of the two-noded line element. The letters P, I, F and S written in superscript after Π represent "Plastic deformation", "Incompressibility", "Friction" and "Sliding contact" respectively.

4.3 Plastic Deformation Energy Term

By using the finite element procedure, the first derivative of the velocity field in a triangular element e can be represented as

$$\frac{\partial \dot{u}_i}{\partial x_j} = \sum_{k=1}^{3} v_i^k \frac{\partial \phi^k}{\partial x_j} \tag{4.7}$$

The strain rate tensor

$$\dot{\epsilon}_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \tag{4.8}$$

is thus expressed in terms of the shape functions by

$$\dot{\epsilon}_{ij} = \frac{1}{2} \sum_{k=1}^{3} \left(v_i^k \frac{\partial \phi^k}{\partial x_j} + v_j^k \frac{\partial \phi^k}{\partial x_i}\right) \tag{4.9}$$

The calculation of the derivatives of the shape functions is given in appendix (A-1). Therefore, the plastic deformation energy term Π^P for a plane strain upper bound problem can be expressed as

$$\begin{aligned}
\Pi^P &= Y^U \sum_{e=1}^{E} \int_{A_e} \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}\, dA \\
&= \frac{Y^U}{2} \sum_{e=1}^{E} \left\{ \int_{A_e} \sqrt{\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{3} \left(v_i^k \frac{\partial \phi^k}{\partial x_j} + v_j^k \frac{\partial \phi^k}{\partial x_i} \right)^2}\, dA \right\} \\
&= \frac{Y^U}{2} \sum_{e=1}^{E} \left\{ A_e \sqrt{\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{3} \left(v_i^k \frac{\partial \phi^k}{\partial x_j} + v_j^k \frac{\partial \phi^k}{\partial x_i} \right)^2} \right\}
\end{aligned} \tag{4.10}$$

It should be noted that the first derivative of the linear shape function ϕ^k is a constant, and the integration in equation (4.10) can thus be easily calculated.

To find the minimum plastic deformation energy Π^P, it is necessary to calculate its derivative with respect to each velocity component v_a^b for $1 \leq a \leq 2$ and $b = 1, 2, ...N$.

$$\begin{aligned}
\frac{\partial \Pi^P}{\partial v_a^b} &= Y^U \frac{\partial}{\partial v_a^b} \sum_{e=1}^{E} \int_{A_e} \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}\, dA \\
&= Y^U \sum_{e=1}^{E} \int_{A_e} \frac{1}{2\sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}} \frac{\partial}{\partial v_a^b} (\dot{\epsilon}_{ij}\dot{\epsilon}_{ij})\, dA
\end{aligned} \tag{4.11}$$

From

$$\begin{aligned}
\frac{\partial}{\partial v_a^b} \left(\frac{\partial \dot{u}_i}{\partial x_j} \right) &= \frac{\partial}{\partial v_a^b} \sum_{k=1}^{3} v_i^k \frac{\partial \phi^k}{\partial x_j} \\
&= \begin{cases} \frac{\partial \phi^b}{\partial x_j} & \text{if } a = i, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned} \tag{4.12}$$

leading to

$$\frac{\partial}{\partial v_a^b}(\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}) = 2\dot{\epsilon}_{ij} \frac{\partial}{\partial v_a^b} \dot{\epsilon}_{ij}$$

$$
\begin{aligned}
&= 2\sum_{i=1}^{2}\sum_{j=1}^{2}\left\{\left[\frac{1}{2}\sum_{k=1}^{3}\left(v_i^k\frac{\partial\phi^k}{\partial x_j}+v_j^k\frac{\partial\phi^k}{\partial x_i}\right)\right]\right.\\
&\quad\left.\times\left[\frac{1}{2}\frac{\partial}{\partial v_a^b}\sum_{k=1}^{3}\left(v_i^k\frac{\partial\phi^k}{\partial x_j}+v_j^k\frac{\partial\phi^k}{\partial x_i}\right)\right]\right\}\\
&= \sum_{l=1}^{2}\sum_{k=1}^{3}\left(v_a^k\frac{\partial\phi^k}{\partial x_l}+v_l^k\frac{\partial\phi^k}{\partial x_a}\right)\frac{\partial\phi^b}{\partial x_l} \qquad (4.13)
\end{aligned}
$$

Hence, equation (4.11) becomes

$$
\frac{\partial\Pi^P}{\partial v_a^b} = Y^U\sum_{e=1}^{E}\int_{A_e}\frac{1}{2\sqrt{\dot\epsilon_{ij}\dot\epsilon_{ij}}}\left[\sum_{l=1}^{2}\sum_{k=1}^{3}\left(v_a^k\frac{\partial\phi^k}{\partial x_l}+v_l^k\frac{\partial\phi^k}{\partial x_a}\right)\frac{\partial\phi^b}{\partial x_l}\right]dA \qquad (4.14)
$$

The expression $\sqrt{\dot\epsilon_{ij}\dot\epsilon_{ij}}$ depends on the values $\{v_i^j\}$. An iterative calculation procedure starting with a guess value for each element is used to obtain the solution. For an element e, the calculated $\sqrt{\dot\epsilon_{ij}\dot\epsilon_{ij}}$, i.e. the plastic energy terms without the constant Y^U, is defined by

$$
W_e = \frac{1}{2}\sqrt{\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{3}\left(v_i^k\frac{\partial\phi^k}{\partial x_j}+v_j^k\frac{\partial\phi^k}{\partial x_i}\right)^2} \qquad (4.15)
$$

Thus, the first derivative of the plastic deformation energy Π^P for a velocity component v_a^b is obtained by

$$
\begin{aligned}
\frac{\partial\Pi^P}{\partial v_a^b} &= Y^U\sum_{e=1}^{E}\int_{A_e}\frac{1}{2W_e}\left[\sum_{l=1}^{2}\sum_{k=1}^{3}\left(v_a^k\frac{\partial\phi^k}{\partial x_l}+v_l^k\frac{\partial\phi^k}{\partial x_a}\right)\frac{\partial\phi^b}{\partial x_l}\right]dA\\
&= Y^U\sum_{e=1}^{E}\frac{A_e}{2W_e}\left[\sum_{l=1}^{2}\sum_{k=1}^{3}\left(v_a^k\frac{\partial\phi^k}{\partial x_l}+v_l^k\frac{\partial\phi^k}{\partial x_a}\right)\frac{\partial\phi^b}{\partial x_l}\right] \qquad (4.16)
\end{aligned}
$$

The second derivative of Π^P with respect to a velocity component v_c^d in the direction c at the d-th node can be calculated by

$$
\begin{aligned}
\frac{\partial}{\partial v_c^d}\left(\frac{\partial\Pi^P}{\partial v_a^b}\right) &= \frac{\partial}{\partial v_c^d}\left(Y^U\frac{\partial}{\partial v_a^b}\sum_{e=1}^{E}\int_{A_e}\sqrt{\dot\epsilon_{ij}\dot\epsilon_{ij}}\,dA\right)\\
&= \frac{\partial}{\partial v_c^d}\left(Y^U\sum_{e=1}^{E}\int_{A_e}\frac{1}{2\sqrt{\dot\epsilon_{ij}\dot\epsilon_{ij}}}\frac{\partial}{\partial v_a^b}\left(\dot\epsilon_{ij}\dot\epsilon_{ij}\right)\,dA\right)
\end{aligned}
$$

$$= Y^U \sum_{e=1}^{E} \int_{A_e} \frac{-1}{4\left(\sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}\right)^3} \left[\frac{\partial}{\partial v_c^d}(\dot{\epsilon}_{ij}\dot{\epsilon}_{ij})\right]\left[\frac{\partial}{\partial v_a^b}(\dot{\epsilon}_{ij}\dot{\epsilon}_{ij})\right] + \frac{1}{2\sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}} \frac{\partial}{\partial v_c^d}\left[\frac{\partial}{\partial v_a^b}(\dot{\epsilon}_{ij}\dot{\epsilon}_{ij})\right] dA \tag{4.17}$$

Due to

$$\begin{aligned} \frac{\partial}{\partial v_c^d}\left[\frac{\partial}{\partial v_a^b}(\dot{\epsilon}_{ij}\dot{\epsilon}_{ij})\right] &= \frac{\partial}{\partial v_c^d}\left[\sum_{l=1}^{2}\sum_{k=1}^{3}\left(v_a^k\frac{\partial\phi^k}{\partial x_l} + v_l^k\frac{\partial\phi^k}{\partial x_a}\right)\frac{\partial\phi^b}{\partial x_l}\right] \\ &= \frac{\partial\phi^d}{\partial x_a}\frac{\partial\phi^b}{\partial x_c} + \sum_{l=1}^{2}\sum_{k=1}^{3}\delta_{ac}\frac{\partial\phi^k}{\partial x_l} \end{aligned} \tag{4.18}$$

Thus, the second derivative with respect to the velocity component v_c^d is

$$\begin{aligned} \frac{\partial}{\partial v_c^d}\left(\frac{\partial\Pi^P}{\partial v_a^b}\right) = Y^U \sum_{e=1}^{E} A_e \Bigg\{ &\frac{-1}{4W_e^3}\left[\sum_{l=1}^{2}\sum_{k=1}^{3}\left(v_c^k\frac{\partial\phi^k}{\partial x_l} + v_l^k\frac{\partial\phi^k}{\partial x_c}\right)\frac{\partial\phi^d}{\partial x_l}\right] \\ &\times\left[\sum_{l=1}^{2}\sum_{k=1}^{3}\left(v_a^k\frac{\partial\phi^k}{\partial x_l} + v_l^k\frac{\partial\phi^k}{\partial x_a}\right)\frac{\partial\phi^b}{\partial x_l}\right] \\ &+ \frac{1}{2W_e}\left(\frac{\partial\phi^d}{\partial x_a}\frac{\partial\phi^b}{\partial x_c} + \sum_{l=1}^{2}\sum_{k=1}^{3}\delta_{ac}\frac{\partial\phi^k}{\partial x_l}\right)\Bigg\} \end{aligned} \tag{4.19}$$

4.4 Incompressibility Term

The incompressibility conditions are treated by the penalty method. The first derivative of the incompressibility term Π^I in equation (4.6), with respect to the velocity component v_a^b is expressed by

$$\begin{aligned} \frac{\partial\Pi^I}{\partial v_a^b} &= \frac{\partial}{\partial v_a^b}\sum_{e=1}^{E}\int_{A_e}\alpha(\dot{\epsilon}_{ii})^2\,dA \\ &= \sum_{e=1}^{E}\int_{A_e}\alpha\frac{\partial}{\partial v_a^b}(\dot{\epsilon}_{ii})^2\,dA \\ &= \sum_{e=1}^{E}\int_{A_e}2\alpha\dot{\epsilon}_{ii}\frac{\partial}{\partial v_a^b}\dot{\epsilon}_{ii}\,dA \end{aligned} \tag{4.20}$$

From

$$\begin{aligned} \dot{\epsilon}_{ii} &= \sum_{i=1}^{2}\sum_{k=1}^{3} v_i^k \frac{\partial \phi^k}{\partial x_i} \\ \frac{\partial}{\partial v_a^b}\dot{\epsilon}_{ii} &= \frac{\partial \phi^b}{\partial x_a} \end{aligned} \tag{4.21}$$

leading to

$$\begin{aligned} \frac{\partial \Pi^I}{\partial v_a^b} &= \sum_{e=1}^{E}\int_{A_e} 2\alpha \sum_{i=1}^{2}\sum_{k=1}^{3} v_i^k \frac{\partial \phi^k}{\partial x_i}\frac{\partial \phi^b}{\partial x_a}\, dA \\ &= \sum_{e=1}^{E} 2\alpha A_e \sum_{i=1}^{2}\sum_{k=1}^{3} v_i^k \frac{\partial \phi^k}{\partial x_i}\frac{\partial \phi^b}{\partial x_a} \end{aligned} \tag{4.22}$$

Similarly, the second derivative of the incompressibility term with respect to the velocity component v_c^d has the form

$$\begin{aligned} \frac{\partial}{\partial v_c^d}\left(\frac{\partial \Pi^I}{\partial v_a^b}\right) &= \sum_{e=1}^{E}\int_{A_e} 2\alpha \left[\left(\frac{\partial}{\partial v_c^d}\dot{\epsilon}_{ii}\right)\left(\frac{\partial}{\partial v_a^b}\dot{\epsilon}_{ii}\right) + \dot{\epsilon}_{ii}\frac{\partial}{\partial v_c^d}\left(\frac{\partial}{\partial v_a^b}\dot{\epsilon}_{ii}\right)\right] dA \\ &= \sum_{e=1}^{E}\int_{A_e} 2\alpha \frac{\partial \phi^d}{\partial x_c}\frac{\partial \phi^b}{\partial x_a}\, dA \\ &= \sum_{e=1}^{E} 2\alpha A_e \frac{\partial \phi^d}{\partial x_c}\frac{\partial \phi^b}{\partial x_a} \end{aligned} \tag{4.23}$$

4.5 Friction Term

The Tresca friction model is used to express the magnitude of the frictional stress in terms of the relative sliding velocity and the yield shear stress k. The frictional stress is then given by

$$\tau_i^f = mk\frac{\dot{u}_i^r}{\|\dot{u}_i^r\|} \tag{4.24}$$

where

$$\|\dot{u}_i^r\| = \sqrt{\sum_{i=1}^{2}(\dot{u}_i^r)^2}. \tag{4.25}$$

The frictional energy term is obtained by

$$\begin{aligned}\Pi^F &= \int_{S_f} \tau_i^f \dot{u}_i^r \, dS \\ &= \sum_{h=1}^{H} \int_{L_h} mk \frac{\dot{u}_i^r}{\|\dot{u}_i^r\|} \dot{u}_i^r \, dS \\ &= \sum_{h=1}^{H} \int_{L_h} mk \|\dot{u}_i^r\| \, dS \end{aligned} \tag{4.26}$$

From equation (4.4), equation (4.25) becomes

$$\|\dot{u}_i^r\| = \sqrt{\sum_{i=1}^{2} (\sum_{j=1}^{2} \psi^j v_i^{rj})^2} \tag{4.27}$$

Since the shape function is involved in the root square, it is not convenient to integrate equation (4.27). An approximation without the shape function in the root square may be useful to overcome the difficulty, which is

$$\begin{aligned}\|\dot{u}_i^r\| &= \sqrt{\sum_{i=1}^{2} (\sum_{j=1}^{2} \psi^j v_i^{rj})^2} \\ &\cong \sum_{j=1}^{2} \psi^j \sqrt{\sum_{i=1}^{2} (v_i^{rj})^2} \end{aligned} \tag{4.28}$$

Since the values of relative velocities v_i^{rj} are very close to the values of adjacent nodes, the approximate equation (4.28) is thus reasonable. The friction energy term becomes

$$\Pi^F = \sum_{h=1}^{H} \int_{L_h} mk \sum_{j=1}^{2} \psi^j \sqrt{\sum_{i=1}^{2} (v_i^{rj})^2} \, dS \tag{4.29}$$

The first derivative of Π^F with respect to a nodal velocity component v_a^b is thus expressed by

$$\frac{\partial \Pi^F}{\partial v_a^b} = \sum_{h=1}^{H} \int_{L_h} mk \frac{\partial}{\partial v_a^b} \left(\sum_{j=1}^{2} \psi^j \sqrt{\sum_{i=1}^{2} (v_i^{rj})^2} \right) \, dS$$

$$= \sum_{h=1}^{H} \int_{L_h} mk\psi^b \frac{v_a^{rb}}{\sqrt{\sum_{i=1}^{2}(v_i^{rb})^2}}\, dS$$
$$= \sum_{h=1}^{H} \frac{mkL_h}{2} \frac{v_a^{rb}}{V^b} \tag{4.30}$$

where

$$V^b = \sqrt{\sum_{i=1}^{2}(v_i^{rb})^2}$$

which is the relative speed of the node b. The integration of the shape function ψ^b in equation (4.30) uses

$$\int_{L_h} \psi^b\, dS = \frac{L_h}{2} \tag{4.31}$$

Similarly, the second derivative of Π^F with respect to a nodal velocity component v_c^d is written by

$$\begin{aligned}
\frac{\partial}{\partial v_c^d}\left(\frac{\partial \Pi^F}{\partial v_a^b}\right) &= \sum_{h=1}^{H} \frac{mkL_h}{2} \frac{\partial}{\partial v_c^d}\left(\frac{v_a^{rb}}{\sqrt{\sum_{i=1}^{2}(v_i^{rb})^2}}\right) \\
&= \sum_{h=1}^{H} \frac{mkL_h}{2} \left\{ \left(\frac{\partial}{\partial v_c^d} v_a^{rb}\right) \left[\sum_{i=1}^{2}(v_i^{rb})^2\right]^{-\frac{1}{2}} \right. \\
&\quad \left. + v_a^{rb}\left(\frac{-1}{2}\right) \left[\sum_{i=1}^{2}(v_i^{rb})^2\right]^{-\frac{3}{2}} (2v_c^{rd}) \right\} \\
&= \sum_{h=1}^{H} \frac{mkL_h}{2} \left(\frac{\delta_{ac}\delta_{bd}}{V^b} - \frac{v_a^{rb}}{v_c^{rd}}\right)
\end{aligned} \tag{4.32}$$

4.6 Sliding Contact Constraints Term

The sliding contact constraints are treated by the penalty term Π^S, i.e. equation (4.5). Its first derivative with respect to a velocity component v_a^b is expressed by

$$\begin{aligned} \frac{\partial \Pi^S}{\partial v_a^b} &= \frac{\partial}{\partial v_a^b}\left(\beta \sum_{j=1}^{H} (v_i^{rj} n_i^j)^2\right) \\ &= 2\beta (v_i^{rb} n_i^b) n_a^b \end{aligned} \tag{4.33}$$

The second derivative with respect to a velocity component v_c^d is obtained by

$$\frac{\partial}{\partial v_c^d}\left(\frac{\partial \Pi^S}{\partial v_a^b}\right) = 2\beta n_c^d n_a^b \tag{4.34}$$

4.7 Summary

The upper bound theory is formulated in a discrete domain by using three-noded triangle elements for the workpiece and two-noded line elements for modelling friction effects. These formulations have the benefit of reducing the complexity of the integration before they are implemented into a computer code for the forging simulation. The energy dissipation power of the forging process Π and its derivative with respect to a velocity component v_a^b in the direction a at the node b can be expressed as follows.

$$\begin{aligned} \Pi &= \Pi^P + \Pi^I + \Pi^F + \Pi^S \\ \frac{\partial \Pi}{\partial v_a^b} &= \frac{\partial}{\partial v_a^b}\left(\Pi^P + \Pi^I + \Pi^F + \Pi^S\right) \end{aligned} \tag{4.35}$$

To solve a set of non-linear equations,

$$\frac{\partial \Pi}{\partial v_a^b} = 0 \tag{4.36}$$

resulting from the minimisation of the energy, the Newton-Raphson method with a line search approach is used to determine the optimum velocity field in the deformed workpiece. The method and other numerical treatment are described in the next chapter.

Chapter 5

Numerical Treatment and Structure of Computer Code

5.1 Introduction

The Newton-Raphson method with a line search technique had been used in TEUBA for solving the system equations before this study started. Therefore, to improve the initial estimate of velocity for each new increment, a time step reduction technique has been proposed and implemented into the program. Some regions of a deforming workpiece with low velocity or even without deformation, causing numerical problems, have been explored. The value for the penalty factors and the effect of the friction factor on forging load are investigated by different mesh densities and the number of increments. Flow charts of TEUBA are also presented in this chapter to make a clear structure of the simulation.

5.2 Newton-Raphson Method with Line Search Technique

The upper bound theory with the finite element technique models the forging problem in a discrete domain using a set of non-linear equations (4.36) of the form

$$\mathbf{F}(\mathbf{v}) = \mathbf{0} \tag{5.1}$$

or

$$F_i(v_1, v_2, v_3, ..., v_w) = 0 \quad i = 1, 2, ..., w. \tag{5.2}$$

where $\mathbf{v}$ denotes the entire vector of velocities v_i and $\mathbf{F}$ denotes the entire vector of functions F_i containing the derivatives of the energy with respect to each velocity component in vector $\mathbf{v}$. w is the number of variables, i.e. the number of unknown velocity components. These equations are non-linear and a multidimensional root-finding scheme is thus required. The Newton-Raphson method of solving non-linear equations from an initial guess is commonly used for metal forming problems due to its large convergence range and relatively fast convergence rate [90, 98, 108, 109].

In the neighbourhood of $\mathbf{v}$, i.e. $\mathbf{v}$ is increased by a small increment $\delta\mathbf{v}$, each of the functions F_i can be expanded in a Taylor series

$$F_i(\mathbf{v} + \delta\mathbf{v}) = F_i(\mathbf{v}) + \sum_{j=1}^{w} \frac{\partial F_i}{\partial v_j}\delta v_j + O(\delta\mathbf{v}^2) \tag{5.3}$$

where O is the Big Oh notation. The matrix of partial derivatives in equation

(5.3) is the Jacobian matrix **J**; i.e.

$$J_{ij} = \frac{\partial F_i}{\partial v_j}$$

Equation (5.3) in matrix notation becomes

$$\mathbf{F}(\mathbf{v} + \delta\mathbf{v}) = \mathbf{F}(\mathbf{v}) + \mathbf{J} \cdot \delta\mathbf{v} + O(\delta\mathbf{v}^2) \tag{5.4}$$

By neglecting terms of order $\delta\mathbf{v}^2$ and higher and setting $\mathbf{F}(\mathbf{v} + \delta\mathbf{v}) = 0$, a set of linear equations for the $\delta\mathbf{v}$ that move each function F_i closer to zero is obtained by

$$\mathbf{J} \cdot \delta\mathbf{v} = -\mathbf{F} \tag{5.5}$$

Matrix equation (5.5) can then be solved by a linear equation solver, such as LU decomposition [103, 104], and thus

$$\delta\mathbf{v} = -\mathbf{J}^{-1} \cdot \mathbf{F} \tag{5.6}$$

$\delta\mathbf{v}$ is called the Newton step. The LU decomposition factorises a matrix to be the product of **LU** of a lower triangular matrix **L** in which the leading diagonal elements are unity and upper triangular matrix **U**. The application of Newton-Raphson method to equation (5.1) leads to an iterative procedure with respect to **v**

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \lambda\delta\mathbf{v}_i$$

where λ is called the step length. The value of the step length λ may be determined by a strategy which requires that the step decreases $|\mathbf{F}|^2$ ($|\mathbf{F}|^2 = \mathbf{F} \cdot \mathbf{F}$). This is the same requirement for minimising

$$f = \frac{1}{2}\mathbf{F} \cdot \mathbf{F}$$

Note that the Newton step, i.e. $\delta \mathbf{v}$, is a descent direction for f:

$$\nabla f \cdot \delta \mathbf{v} = (\mathbf{F} \cdot \mathbf{J}) \cdot (-\mathbf{J}^{-1}\mathbf{F}) = -\mathbf{F} \cdot \mathbf{F} < 0 \tag{5.7}$$

However, a full Newton step, i.e. $\lambda = 1$, may not reduce f. A check is carried out at each iteration to ensure that the proposed step reduces f. If the step is too large, the value of λ is reduced by backtracking along the Newton direction until an acceptable value is found. A *line search* technique [104, 107] which determines the minimum point on a given line may be used to find an approximate value of λ. The technique described as follows.

The new value of f may be defined by a function g with the variable λ:

$$g(\lambda) \equiv f(\mathbf{v}_{i+1}) = f(\mathbf{v}_i + \lambda \delta \mathbf{v})$$

so that

$$g'(\lambda) = \nabla f \cdot \delta \mathbf{v}$$

If the backtrack is needed, the λ is chosen by minimising the function g with the most current information. If the first step which always starts the full Newton step ($\lambda = 1$) is not acceptable, three values, $g(0)$, $g'(0)$ and $g(1)$, are available to model $g(\lambda)$ as a quadratic curve. The curve can then be used to find a minimum of λ. On the second and subsequent backtracks, the function $g(\lambda)$ is modeled as a cubic curve and thus a new minimum of λ can be determined for the current information. This technique for choosing the step length is called *line search.*

The Newton step is a descent direction for f as shown in equation (5.7), and it is thus guaranteed by using *line search* to find an acceptable step length. Finally, the Newton-Raphson method with the line search technique minimises f by taking

Newton steps designed to bring $\mathbf{F}$ to zero, i.e. to solve velocity $\mathbf{v}$ for $\mathbf{F}$.

The iteration is terminated when

$$\| \mathbf{F}(\mathbf{v}_i, \lambda) \| < e_e \tag{5.8}$$

where e_e is set to 5×10^{-6}.

5.3 Time Step Reduction Technique

The Newton-Raphson method for solving non-linear equations requires an estimate of the nodal velocities for the first iteration. A linear interpolation scheme which assigns a velocity to each node in terms of its distance from the die surface is used to guess the initial velocities. Equation (5.1) is then solved by a direct method which uses a linear equation solver, LU decomposition in TEUBA's case, to reduce numerical errors before the Newton-Raphson method with the line search technique takes over to solve the following iterations until convergence has been reached. In some cases, the Jacobian matrix $\mathbf{J}$ is not positive definite in a Newton-Raphson iteration. This condition causes a numerical problem for the matrix inversion in equation (5.6) and the Newton step $\delta\mathbf{v}$ cannot be found. For such cases, the direct method is used in that iteration and Newton-Raphson method is applied again for the rest of iterations. If both methods fail to solve the problem, a warning message is given and a reducing time step technique developed in this research is applied. The technique is described as follows.

The change of the velocity field in a deforming workpiece is continuous. This fact induces an idea for overcoming the difficulty of finding a convergent velocity field during the simulation in which the forging process is a sequential quasi-static

limit analysis and updates the deformation in a small time step.

The difference of the velocity between two increments should be very small in most cases. Figure 5.1 shows the velocity at point 1, V_1, is close to that at point 2, V_2; i.e. the difference of two vectors, V_{21}, is small. Therefore, V_1 could be a good guess of the initial velocity to calculate the velocity at point 2. However, the velocity may have a siginificant change after a time step, e.g. at the following point 3, V_3, in Figure 5.1. V_2 may not be a good guess of the initial velocity for the calculation of V_3. If the solver fails to solve the velocity for the new increment, the time step is reduced; i.e. the point 3′ is a new deformed position instead of the point 3. The deformation of the workpiece is smaller and closer to that in a normal case; i.e. at point 3. This time step reduction technique thus provides a better guess of the initial velocity for the following new increment. The technique has been implemented into TEUBA to overcome numerical problems which may be caused by a poor initial guess of velocity from the previous increment.

A technique combined with a function, such as a polynomial, to model the variation of the velocity during the deformation could also be used to guess the initial velocity for a new increment. This may be of interest in future studies.

5.4 Treatment of Low Velocity Zone

In the upper bound theory with the finite element procedure, the entire workpiece is assumed to deform plastically. However, some regions with low velocity or even without deformation may exist. This situation will cause numerical problems with infinite terms since the term $W_e = \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}}$, i.e. equation (4.15), in the denominator of the system equations obtained by differentiating the functional,

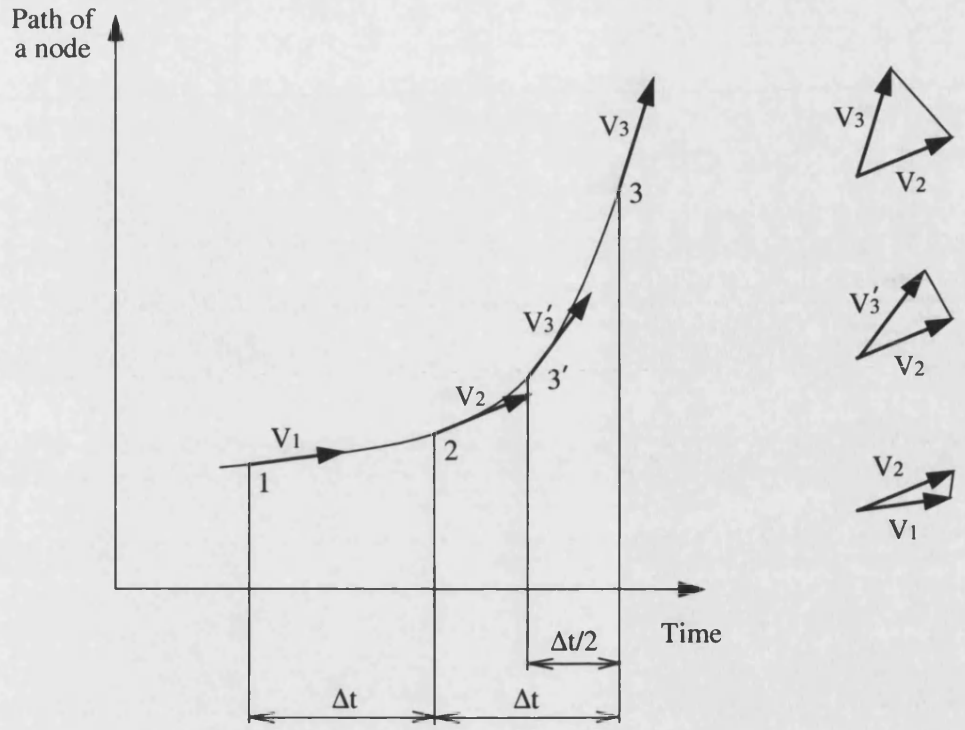


Figure 5.1: Reducing time step technique.

is close to zero. This numerical problem is solved by assigning a small value to the effect strain rate which may be two orders of magnitude less than the average effective strain rate [110, 111]. In this thesis, the term W_e is set as 1×10^{-4} to avoid the ill-conditioned numerical problem for the element with the low velocities.

5.5 Structure of Simulation

5.5.1 Flowchart of Data Transport

The transport of data and files is presented by a flowchart in Figure 5.2. The geometry of the desired forging shape and tool surfaces is constructed by a commercial modeller, CADfix [112], (former FAM—Field Analysis Modeller) which is also able to mesh the geometry of models. A Tcl/Tk script [113] **CAD-**

fix2TEUBA has been written and implemented in CADfix in order to execute the meshing and to export FE data in ASCII format, including the information about the coordinates of nodes and the connectivity of elements. The data is then read by TEUBA to run the simulation of forging process in forward or reverse mode. The re-meshing procedure may be implemented in TEUBA's solver in a commercial package of the FE simulation. However, the re-meshing engine has not been developed, and thus the decision of the re-mesh is made by checking the quality of elements or alternatively a visual evaluation. If such re-meshing procedure is needed, the boundary of the deformed workpiece is then exported by a function of TEUBA to an ASCII file which can be read by a Tcl/Tk script **LoadBSH** into CADfix and re-meshed by another Tcl/Tk script **Re-Mesh**. However, the distance between nodes constructing the boundary of the deformed workpiece may be too small which results in a numerous number of elements after the re-meshing task is carried out. In such cases, the geometry of the workpiece is constructed manually according to the exported information. The above procedure is repeated until the simulation for this forging stage is finished.

5.5.2 Flowchart of TEUBA

The flowchart of the simulation procedure is shown in Figure 5.3. After the FE models of the workpiece and dies are read into TEUBA, it may be necessary to set the yield stress, the friction factor and the simulation mode, i.e. the reverse or forward mode, to run the simulation. An initial velocity field is guessed by the linear interpolation scheme described in Section 5.3. The deforming workpiece and dies are updated in a small time step according to the solved velocity field until the required target distance is reached. The results of the simulation may be recorded if this is needed.

A Graphic User Interface (GUI) based on Motif [114] has been developed on the X Window System to operate the transport of data and to set the parameters of the simulation. A render scene showing the FE models is also integrated with the GUI using Open Inventor [115, 116] which is a library of objects that can be used, modified and extended to meet new needs. The details of using TEUBA can refer to *TEUBA: User and Developer Documentation* [100, 101].

5.5.3 Flowchart of Solver

The plastic deformation energy is calculated by the direct solver for the first iteration in each increment. The following iterations are solved by the Newton-Raphson solver until the velocity field converges. The compressibility of the volume is checked in each iteration. If the incompressibility is not held, the value of the incompressibility penalty, α, will be increased. To avoid instability in the calculations in which the value is too high after several adjustments, the maximum value is set to 1×10^{-6}. A similar procedure is also carried out for the value of the normal penalty, β, which controls the penetration of the node into the die. Table 5.1 shows default numerical parameters for the simulation of TEUBA.

The Newton-Raphson solver may encounter a numerical problem because of ill-conditions. In such cases, the direct solver is used to continue the calculation and swap back to the Newton-Raphson solver for following iterations. The number of failures of the Newton-Raphson solver will be reset to zero if the solver successfully continues calculation. If the number of failures exceeds the maximum number of failures, the technique of reducing time step as described in Section 5.3 is then performed. To avoid large computation times, the maximum number of time step reductions and the maximum number of iterations for this increment

are constrained. Thus, an approximate (i.e. non-converged) velocity may be accepted in some cases and a warning is received from the solver. Finally, the solved velocity field is used to update the FE models for next increment.

Table 5.1: Numerical parameters

Parameter	Value
Incompressibility penalty factor α	
Initial	1×10^4
Maximum	1×10^6
Normal penalty factor β	
Initial	1×10^4
Maximum	1×10^6
Incompressibility convergence factor	5×10^{-4}
Energy convergence factor	5×10^{-6}
Max. number of Newton's failures	6
Max. number of time step reductions	4
Max. number of iterations	50

5.6 Penalty Factor

The penalty factor is used to hold the incompressibility of the plastic deformation. A very large value of the factor is numerically able to keep a constant volume of the workpiece. However, a roundoff error may occur when the penalty factor is too large. To investigate the change of the volume, different penalty factors were used to run simulations of a simple upsetting forging modelled by different numbers of elements and iterations. The results listed in Table 5.2 show the value of the penalty factor may be located between 1×10^4 and 5×10^5. Interestingly, the volume loss significantly decreases as the number of increments increases. This may be because the workpiece geometry changes between increments are smaller in cases with larger number of increments than those with smaller number of increments. The numerical error of updating the mesh of the workpiece is

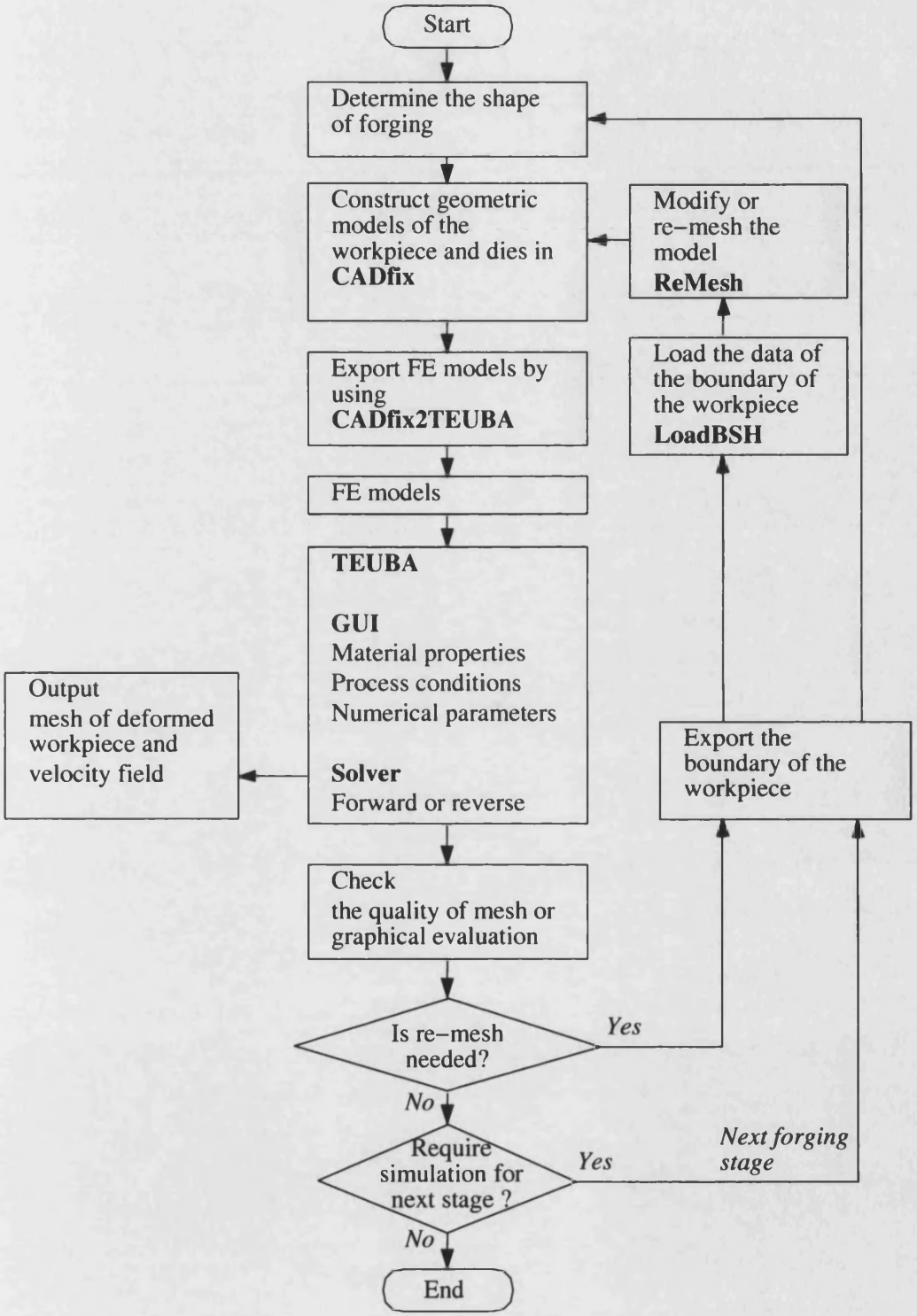


Figure 5.2: Flowchart of data transport.

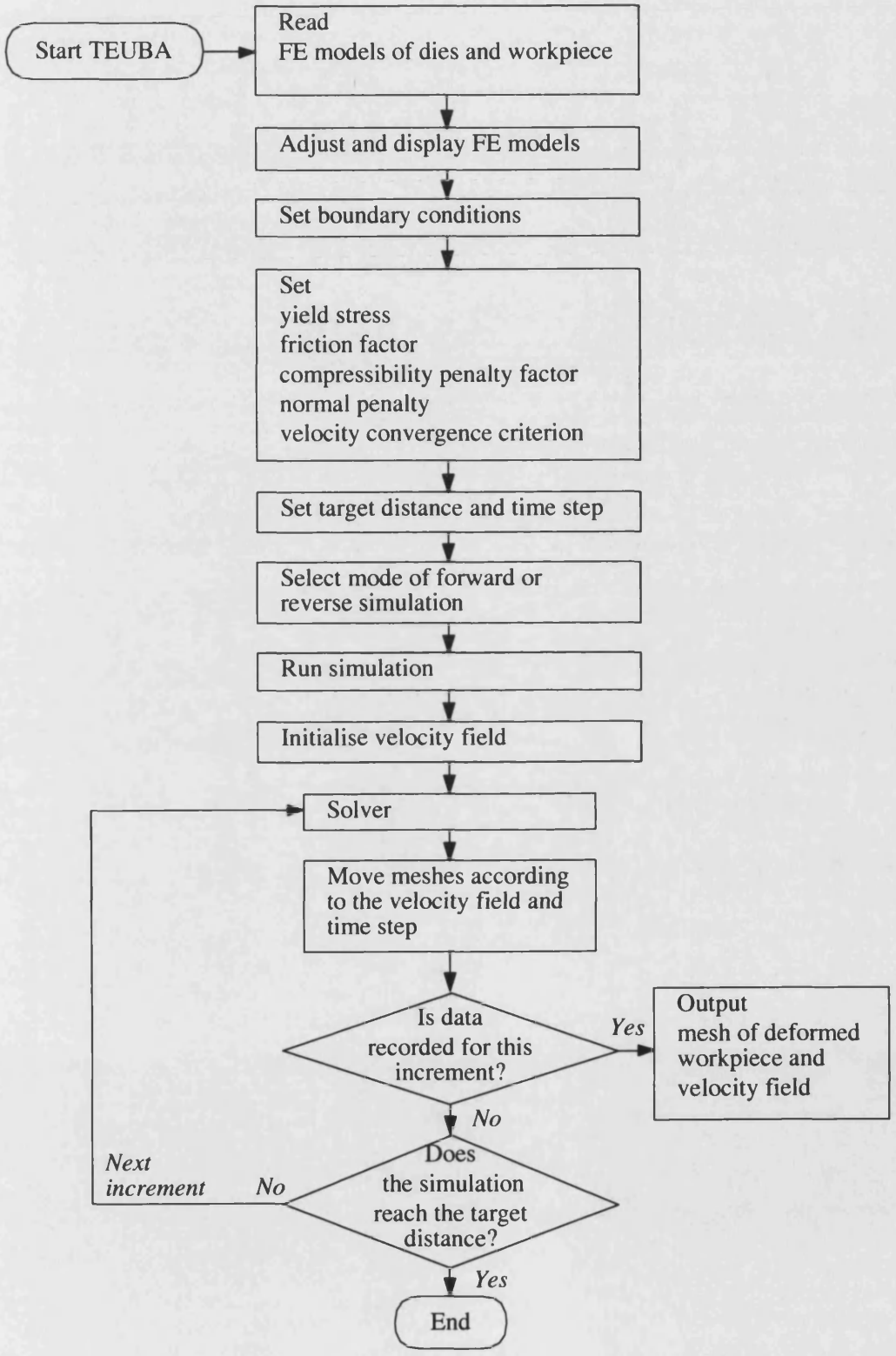


Figure 5.3: Flowchart of TEUBA.

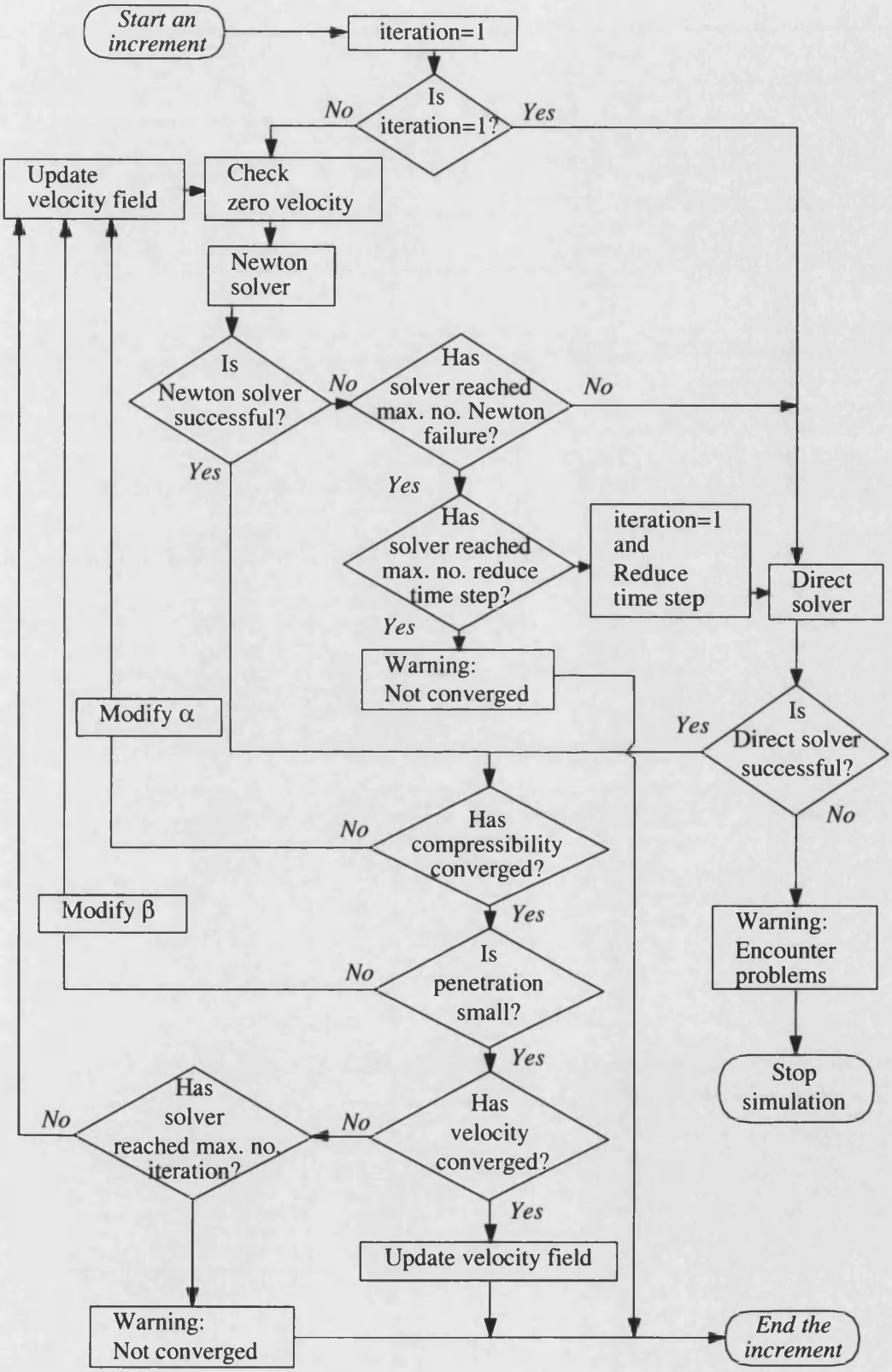


Figure 5.4: Flowchart of TEUBA solver.

therefore reduced as the number of increments is increased. In the test, the volume loss is smaller than 0.18 % in cases with 50 increments. Moreover, from Table 5.2 it also reveals that the effect of the number of elements is not as significant as the number of increments on the volume loss.

Table 5.2: Comparison of volume loss in plane strain simple upsetting tests with different penalty values

		Penalty factor				
		1×10^3	1×10^4	1×10^5	5×10^5	1×10^6
No. elements	No. increments	Vol. loss (%)	Vol. loss (%)	Vol. loss (%)	Vol. loss (%)	Vol. loss (%)
190	10	0.734	0.668	0.663	0.659^{5}	0.661^{10}
	25	0.388	0.303	0.296	0.294^{11}	0.294^{25}
	50	0.254	0.162	0.154	0.153^{27}	E
412	10	0.777	0.707	0.698	0.698^{1}	E
	25	0.407	0.320	0.310	0.309^{0}	E
	50	0.263	0.171	0.161	0.160^{2}	E
678	10	0.748	0.687	0.685	0.686^{0}	E
	25	0.394	0.311	0.305	0.304^{0}	E
	50	0.257	0.167	0.158	0.157^{0}	E

Note:
1. The tests simulate two flat dies compress a rectangular billet with size 10 mm by 10 mm by 25% reduction of the height.
2. The number written in superscript after the value of the volume loss is the number of increments beyond the maximum number of iterations of the calculation. The maximum number is 50 in these tests.
3. E written in the table denotes that the calculation encountered a roundoff error.

5.7 Friction Factor

A constant factor used in the Tresca friction model gives the degree of friction. The factor can be measured by the ring compression test [8, 117, 118]. Its value is between 0.1 and 0.3 for hot forging with lubricant. This section investigates

whether the Tresca friction model can approximate the variation of forging load and deformation as friction factor is varied. An accurate modelling of friction however requires a more complex modelling technique. An investigation of accuracy is not the intention of this approximate modelling approach.

Numerical tests with different friction factors were carried out for a plane strain simple upsetting case which was constructed of 412 elements. Figure 5.5 shows that the barrelling effect becomes more significant as the friction factor increases. Forging load, shown in Figure 5.6, also increases as the friction factor increases from 0.1 to 0.7. The rise of the forging load is however not significant in the cases with the factor 0.9. This phenomenon could result from the fact that the velocity in contacting interfaces is lower in the cases with higher friction.

This simple upsetting forging was also constructed with different numbers of elements to investigate the effect on the modelling of friction. The forging loads induced by friction in these cases are shown in Figure 5.7, 5.8, 5.9 and 5.10. As the number of elements increases, the variation of forging load decreases. The figures also show that a larger friction factor results in a clearer difference of forging load in comparison with the number of elements. This implies that care should be taken when using a large friction factor in TEUBA.

5.8 Summary of Numerical Treatment

The forging process has been modelled by the upper bound method with a finite element formulation. The technique uses linear elements to interpolate the velocity field of deformation. A set of non-linear equations resulting from the minimisation of the dissipation power of the deformation is then obtained and

solved by the Newton-Raphson method with a line search technique.

The performance of the TEUBA was evaluated by Joe Pitt-Frances [105] who compared the computing time of TEUBA's cases with that required by a commercial package, FORGE2. The investigation into the penalty factor for incompressibility and friction factor were carried out in this thesis. These results show that the current formulation can be a rapid approximate approach for modelling forging processes. However, a re-meshing engine has not been developed in TEUBA and the problem of node penetration into dies has not been solved. These two problems reduce the usability of TEUBA and could be future development topics.

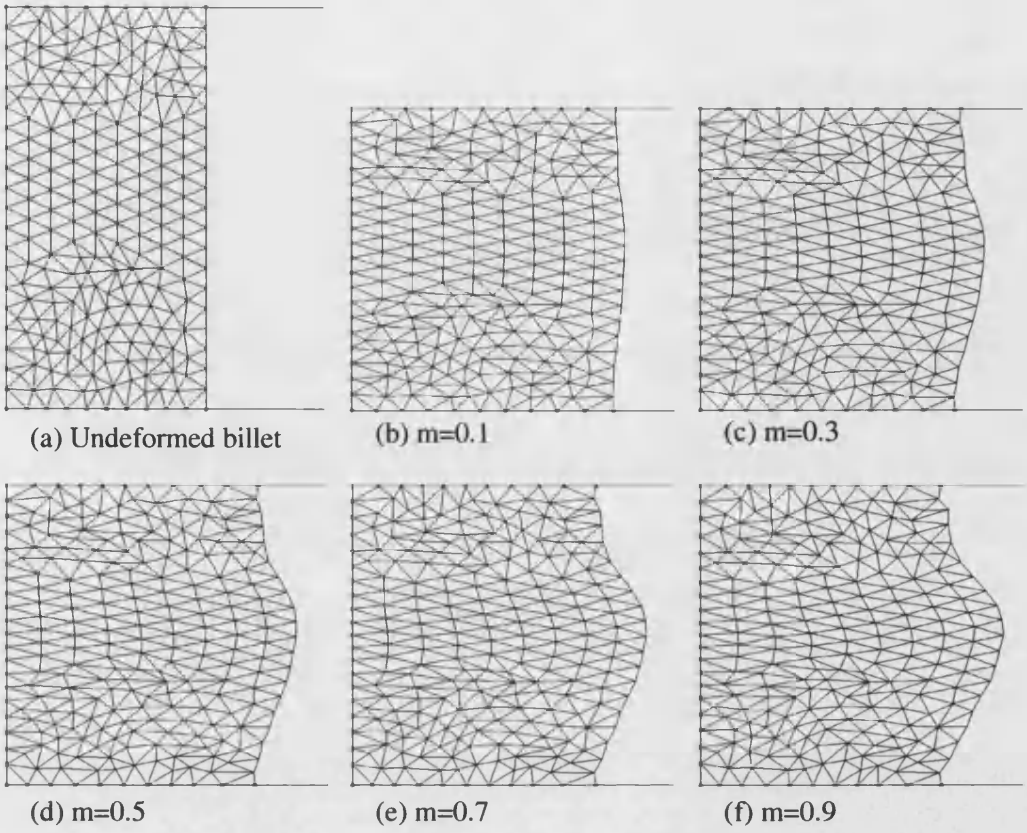


Figure 5.5: Deformation resulting from different friction factors in cases with 412 elements.

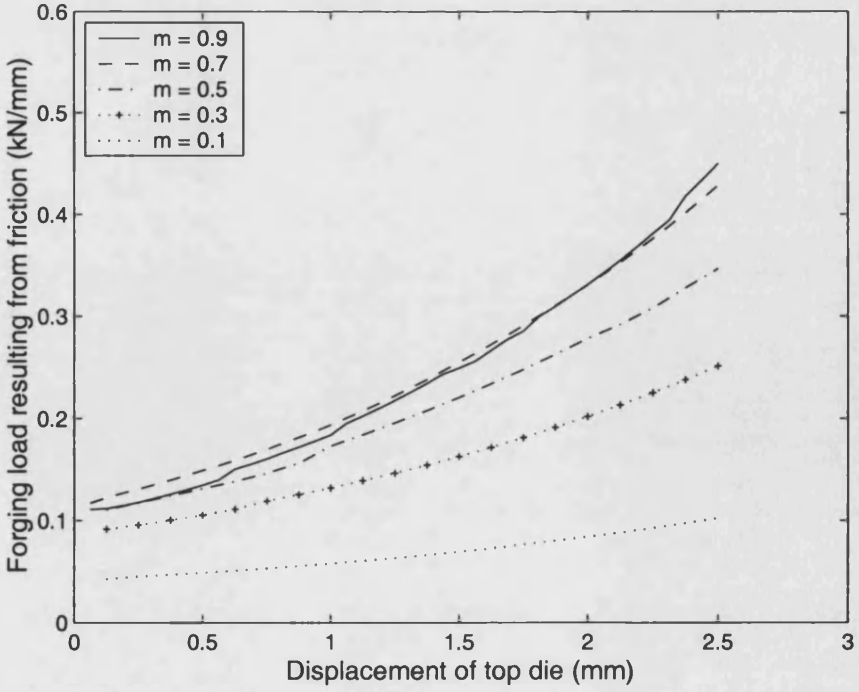


Figure 5.6: Forging load resulting from different friction factors in cases with 412 elements.

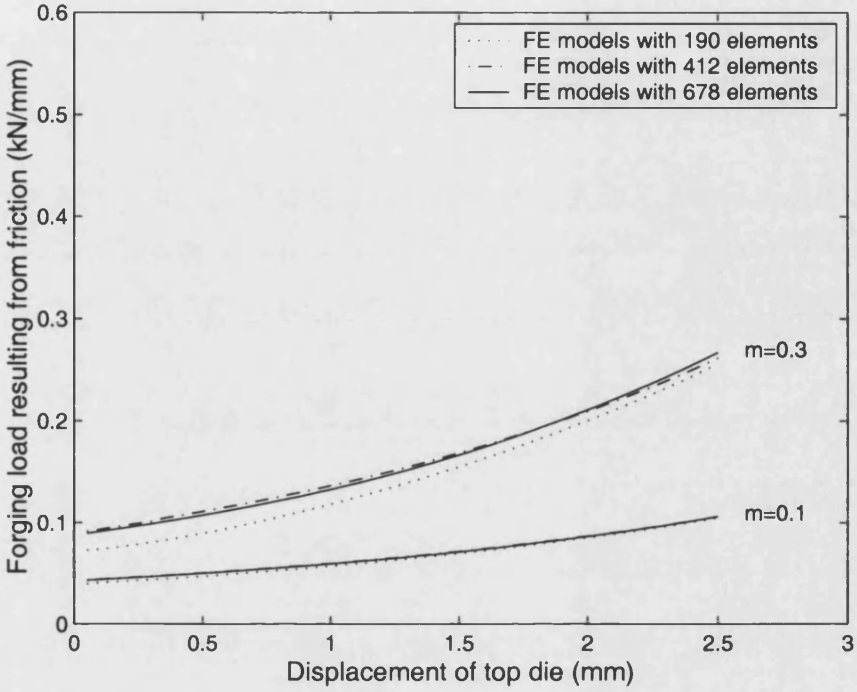


Figure 5.7: Forging load resulting from different friction factors, 0.1 and 0.3, in comparison with number of elements.

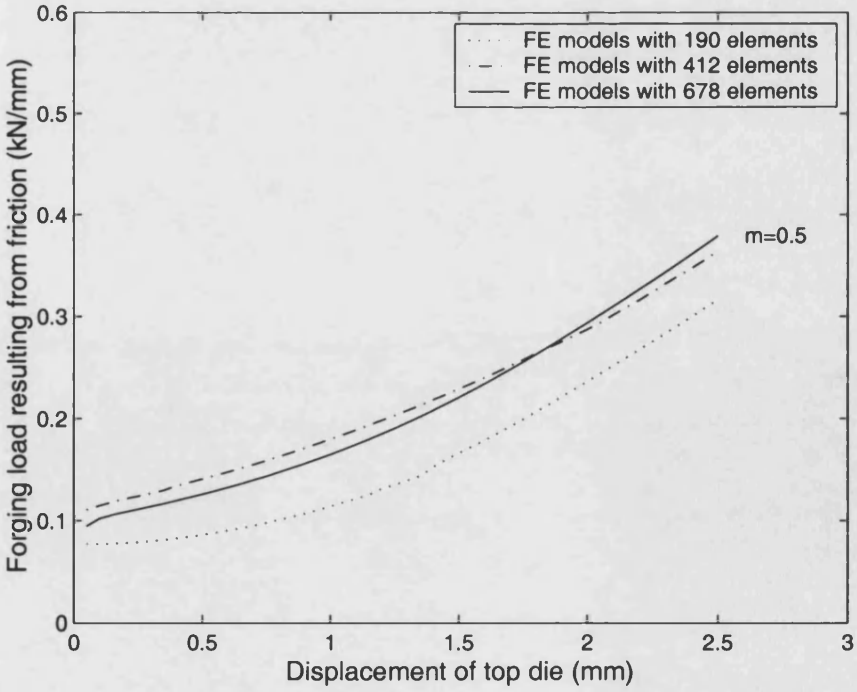


Figure 5.8: Forging load resulting from friction factor 0.5 in comparison with number of elements.

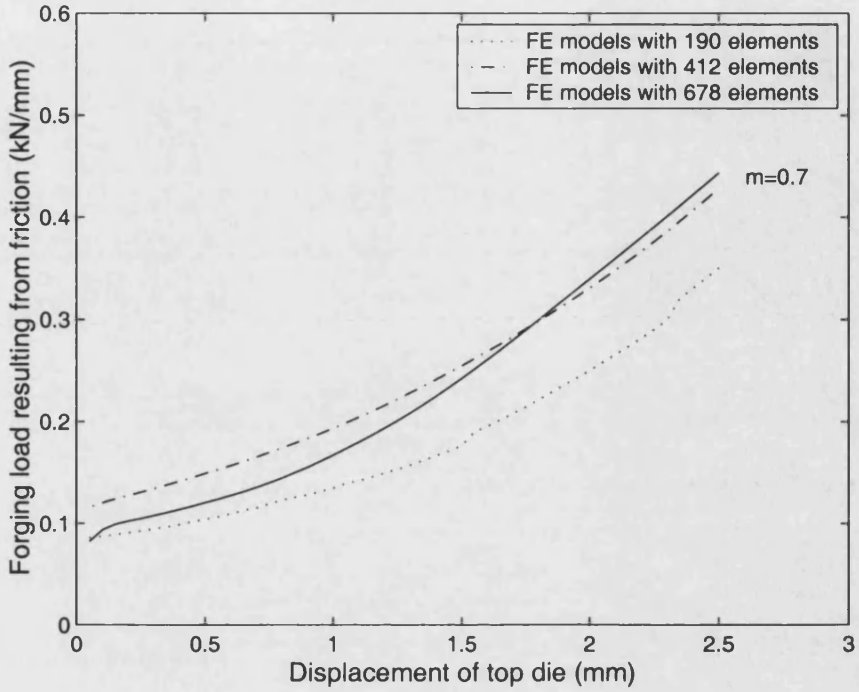


Figure 5.9: Forging load resulting from friction factor 0.7 in comparison with number of elements.

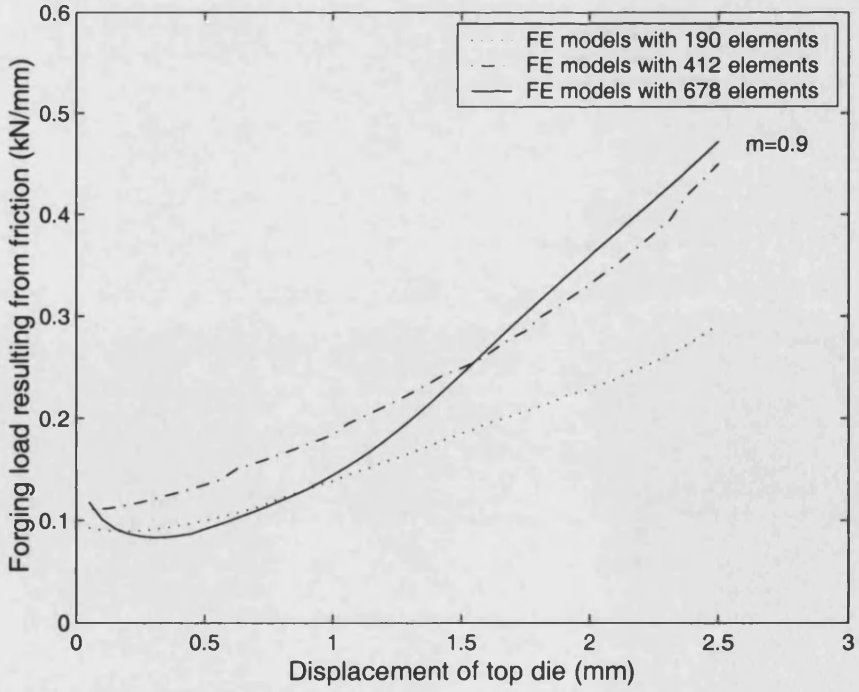


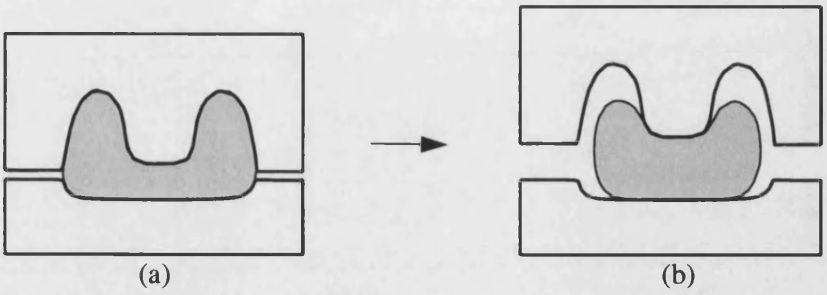
Figure 5.10: Forging load resulting from friction factor 0.9 in comparison with number of elements.

Chapter 6

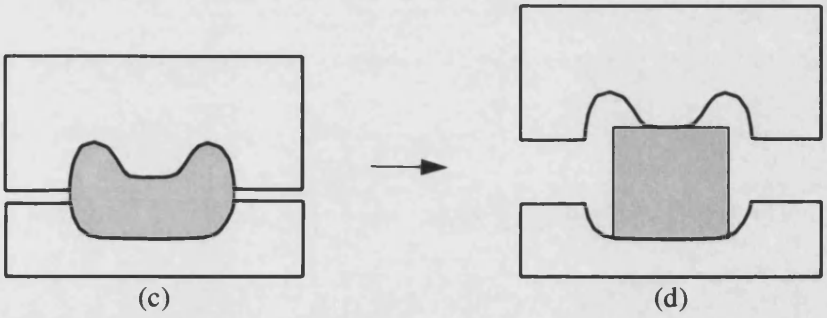
Preform Design Using Reverse Simulation

6.1 Introduction

Reverse simulation has arisen from the concept of simply reversing the flow simulation by starting from the finished shape with the die velocities reversed. With boundary contact areas being released as the die is moved backwards, the simulation predicts an initial billet or a preform when the product cannot be forged in a single forging process. The concept of the reverse simulation for preform design is illustrated in Figure 6.1. According to the behaviour of the workpiece, the forging process is load path dependent, i.e. the deformation depends on the previous loading history. This means that the forging process is not strictly reversible. However, it is possible to model a reversible process which is close to a real forging deformation by making some assumptions. The upper bound theorem with the backwards tracing scheme [56, 59] is employed for this purpose.



First Stage Reverse Simulation
The product shape in (a) is used to design die profiles, and a preform in (b) is predicted after two dies nearly separate from the workpiece.



Second Stage Reverse Simulation
The predicted prefrom in (b) is used to design new die profiles, and a billet in (d) is predicted.

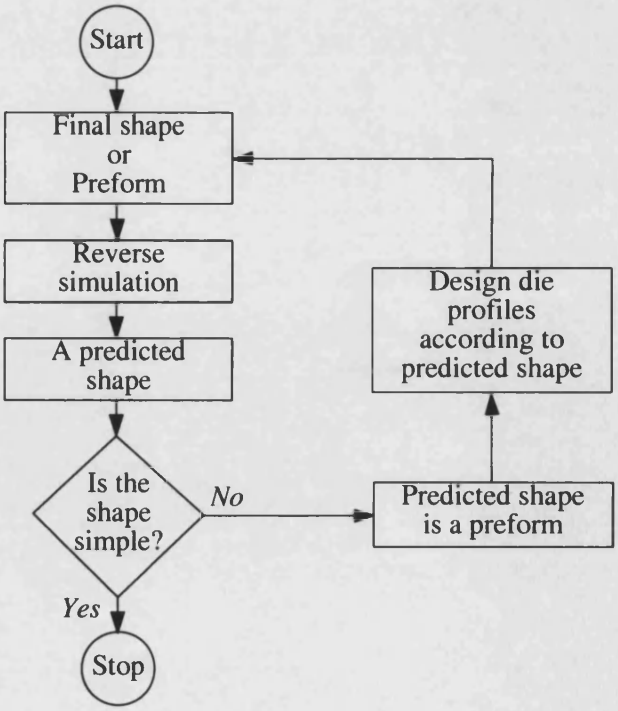


Figure 6.1: Concept and flowchart of using reverse simulation to design preforms.

In forward simulation, the contact conditions used to define the boundary conditions are constantly varying according to the deforming workpiece and the profile of the tools which construct the target shape of the forging process. However, this situation does not exist in reverse simulation. A mechanism of defining the target shape for reverse simulation is therefore needed and then the boundary conditions can be determined. In this thesis, reflecting on the requirement of reducing the geometric complexity of the workpiece during reverse simulation, a concept of using a measure of material distribution is proposed to set a target shape and to calculate the complexity of the workpiece. The target shape is used to identify a contact region which will be released from a die surface and thus the boundary conditions can be established.

To investigate and demonstrate the proposed concept of preform design, the second moment of area is employed to estimate the material distribution in two-dimensional plane strain cases. A rectangle with the closest distribution to the workpiece is set as a target shape of the reverse simulation for the next increment. Possible boundary conditions are then determined by releasing a contact region, which is a finite element node in this thesis, from the die surface according to the target rectangle. The procedure is repeated until all contact regions are released from die surfaces. If the shape so determined is very close to a rectangle, the billet size can readily be established. However, the reversed shape may still be complex. In this case, a preform is needed and the determined shape is taken as the finished product to run another reverse simulation until a simple shape close to a rectangle is found.

6.2 Reversibility — Backwards Tracing Scheme

Loading history determines material properties and varying contact conditions control material flow to construct the deformation of the forging process. These two factors may interact, and are unknown before the desired product is forged. Modelling the forging process in reverse therefore requires such assumptions that treat the process as a loading independent path. The upper bound theorem sequential limit analysis and the backwards tracing scheme could be the basis of such a solution.

Upper Bound Sequential Limit Analysis

The upper bound sequential limit analysis technique employs the rigid-perfectly plastic model for the workpiece material. Strain hardening or softening is neglected in the model and all deformed regions of the workpiece have the same stress value which is a constant. The solution is not affected by the magnitude of the yield stress. Thus, the process is independent of stress and strain status which defines loading history.

Although the upper bound solution only considers the current deforming status, the geometric configuration of the workpiece and the contact conditions are varying during the forging process. As a result, the process is still path dependent. The backwards tracing scheme [56, 59] could be used to solve this problem.

Backwards Tracing Scheme

The backwards tracing scheme was proposed in the early 1980s by Kobayashi et. al. [56, 59]. The concept of the scheme is illustrated in Figure 6.2 and 6.3 At time (increment) t_0, the geometrical configuration $\mathbf{X}_0$ of a deforming workpiece is represented by a point **Q**. During the forward simulation, the point **Q** is obtained

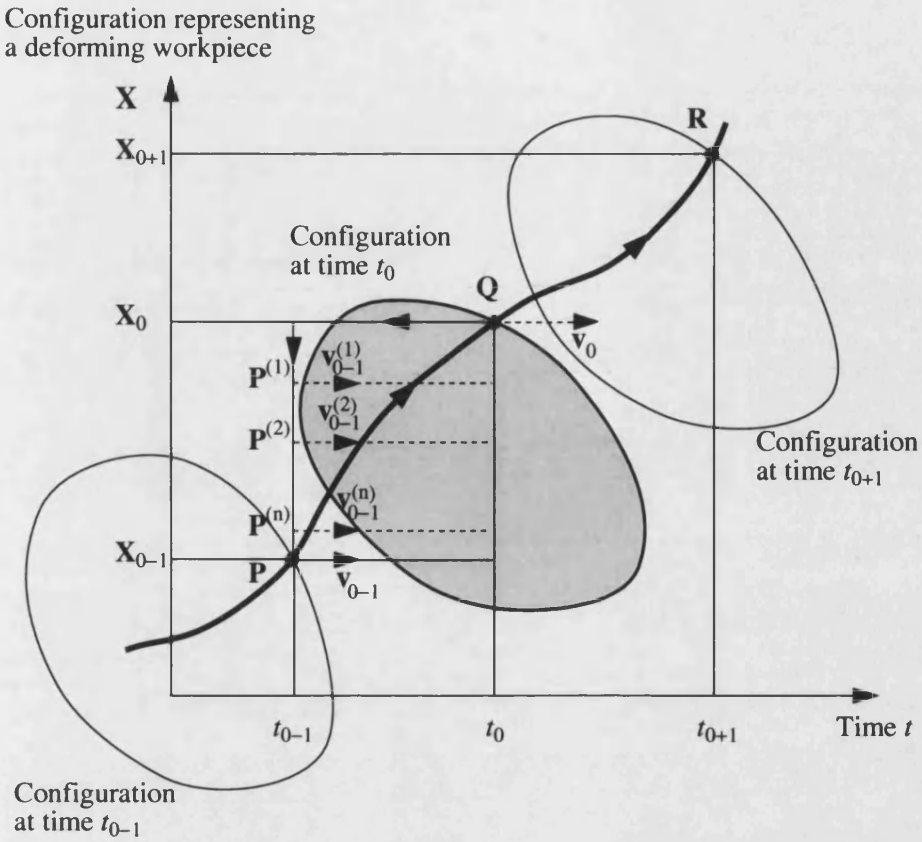


Figure 6.2: Concept of the backwards tracing scheme [56, 59].

by moving from the point $\mathbf{P}$ whose configuration is $\mathbf{X}_{0-1}$ at time t_{0-1} according to the velocity field $\mathbf{v}_{0-1}$ during a time increment $\Delta t = t_0 - t_{0-1}$, and thus

$$\mathbf{X}_0 = \mathbf{X}_{0-1} + \mathbf{v}_{0-1}\Delta t \tag{6.1}$$

Therefore, the problem of the reverse simulation is to determine the velocity field $\mathbf{v}_{0-1}$ based on the known information $\mathbf{X}_0$.

The backwards tracing scheme takes the solution of the forward simulation at point $\mathbf{Q}$. A first estimate of $\mathbf{P}^{(1)}$ can be made according to

$$\mathbf{P}^{(1)} = \mathbf{X}_0 - \mathbf{v}_0\Delta t \tag{6.2}$$

i.e. reversing the configuration of $\mathbf{Q}$ by using the velocity field $\mathbf{v}_0$. Next, the

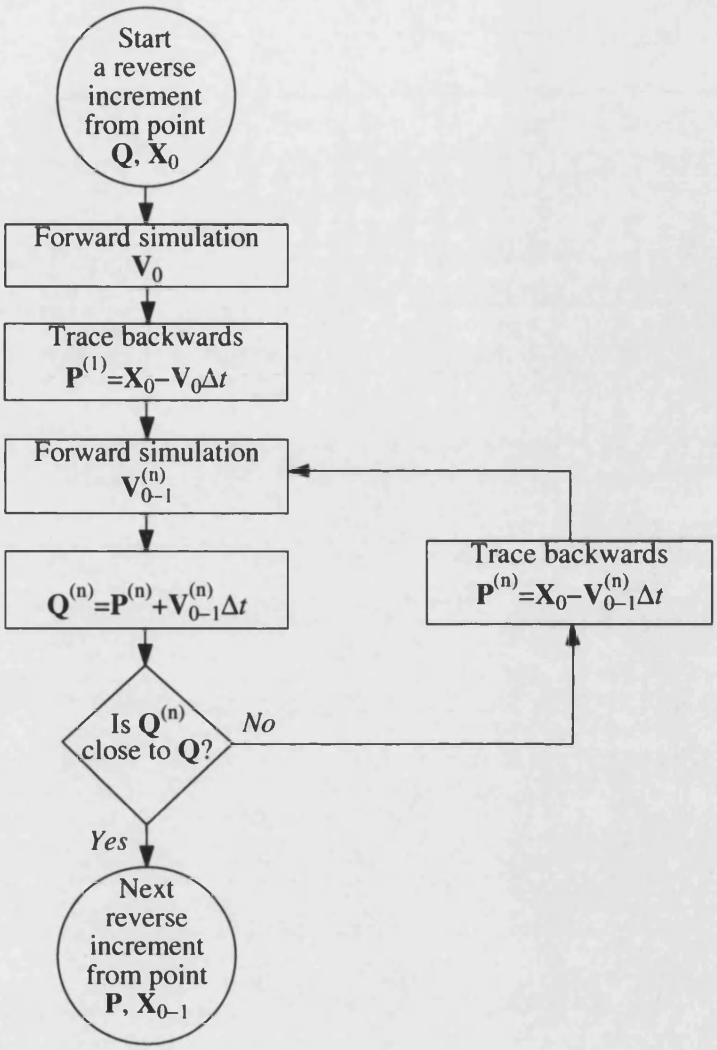


Figure 6.3: A reverse increment using the backwards tracing scheme [56, 59] with known boundary conditions.

velocity filed $\mathbf{v}_{0-1}^{(1)}$ can be solved by the forward simulation based on the $\mathbf{P}^{(1)}$ and used to form $\mathbf{Q}^{(1)}$ by

$$\mathbf{Q}^{(1)} = \mathbf{P}^{(1)} + \mathbf{v}_{0-1}^{(1)}\Delta t \tag{6.3}$$

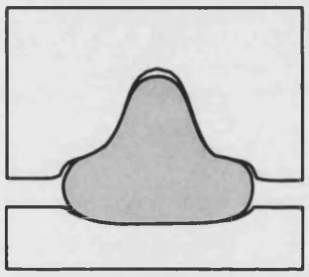
If the $\mathbf{Q}^{(1)}$ is not sufficiently close to the $\mathbf{Q}$, then $\mathbf{P}^{(2)}$ can be estimated by $\mathbf{P}^{(2)} = \mathbf{X}_0 - \mathbf{v}_{0-1}^{(1)}\Delta t$, and the second estimate of the point $\mathbf{Q}^{(2)}$ can be obtained by $\mathbf{Q}^{(2)} = \mathbf{P}^{(2)} + \mathbf{v}_{0-1}^{(2)}\Delta t$. The iteration is carried out until

$$\mathbf{Q}^{(n)} = \mathbf{P}^{(n)} + \mathbf{v}_{0-1}^{(n)}\Delta t \tag{6.4}$$

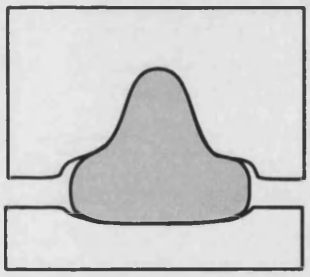
becomes sufficiently close to $\mathbf{Q}$. The criterion for terminating the iteration can be made by

$$||\mathbf{X}_0 - \mathbf{X}_0^{(n)}|| \leq e \tag{6.5}$$

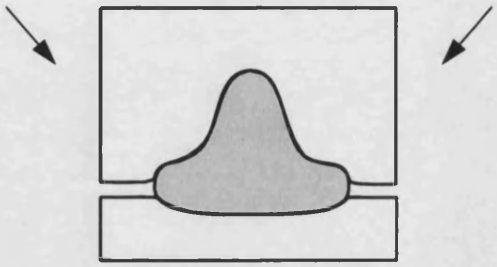
where e is the limit of the difference and is set to 5×10^{-3}, and $\mathbf{X}_0^{(n)}$ represents the geometrical configuration of $\mathbf{Q}^{(n)}$. Finally, the reverse deforming configuration can be defined by the point $\mathbf{P}^{(n)}$. It should be noted that the boundary conditions affect the solution of the velocity field $\mathbf{v}_{0-1}^{(n)}$ and hence the point $\mathbf{P}^{(n)}$. There are infinite numbers of deforming paths from the point $\mathbf{P}$ to $\mathbf{Q}$, which broadly represents a forging process from a billet to a desired shape respectively. Figure 6.4 shows two different contact conditions may achieve complete die filling to form the same desired shape. Therefore, the method for determining the boundary conditions is important to control a backward deformation which aims towards a simple geometric configuration of the workpiece in the reverse simulation.



(a) A free region in top cavity.



(b) A contact region in top cavity.



(c) Two different contact conditions may result in the same shape.

Figure 6.4: Two possible contact conditions resulting from the same geometry of the workpiece.

In this thesis, the formulation for the energy minimisation of the upper bound theorem depends solely on the current geometry and the boundary conditions. The effect of material properties such as strain hardening, strain rate and temperature effects are not taken into account. With the implementation of the backwards tracing scheme, it is possible to predict a deforming path of the workpiece in a reverse direction if the boundary conditions are known.

During forward simulation, the position of the dies and the geometric configuration of the workpiece are used to determine the boundary conditions, i.e. the profiles of the dies construct a target shape for bounding the material flow to form a desired shape. This gives the basis for setting a target shape in the reverse simulation to determine the boundary conditions and thus to control the material flow in reverse to form a simple shape which is a billet or a preform.

6.3 Target Shape and Boundary Conditions

The target shape is defined as the final destination of the deformation in both forward and reverse simulations. The geometric configuration of the desired product, which is the target shape for the forward simulation, is used to examine whether a selected billet and/or a designed preform achieves the requirement for complete die filling. In contrast the reverse simulation starts from a complete die filling condition to search for preforms and a billet without knowing the target shape. It therefore requires the development of a technique for defining the target shape in the reverse simulation. Recognising that preforms including the billet have the common geometric characteristic of simple material distribution compared to the desired shape, a target shape could be defined based on reducing the complexity of the material distribution.

To define a target shape for the reverse simulation, one could for example start from a study of a forging process shown in Figure 6.5(a). In the figure, shape **C** is a preform for shape **D**, and also represents a reversed deformation. Comparing the two shapes in Figure 6.5(b) shows that some contact regions of shape **D** locate on the outside of the profile of shape **C**. If the outside regions of shape **D**, i.e. regions α, β and γ, are released to determine new boundary conditions for a reverse increment of the simulation, it could result in a shape which is close to shape **C**. Thus, the reverse simulation aims towards a simple shape. On the other hand, releasing inside regions, which tends to increase the complexity of shape **D**, is unlikely to predict a shape close to the reversed shape **C**.

Next, the location of the target shape, which is the profile of shape **C** in this example, should be defined because it determines outside contact regions. Figure 6.5(c) shows different outside contact regions are determined by different locations of shape **C**.

Moreover, from the material distribution point of view, a "simplest" billet shape **A** is changed into the "most complex" desired shape **D** after the forging process. Shape **C** is one of the intermediate "complex" shapes between shape **A** and **D** in the deforming process. Therefore, the development of defining an unknown shape **C**, representing the target shape in reverse simulation, can be based on reducing the complexity of the desired shape **D**.

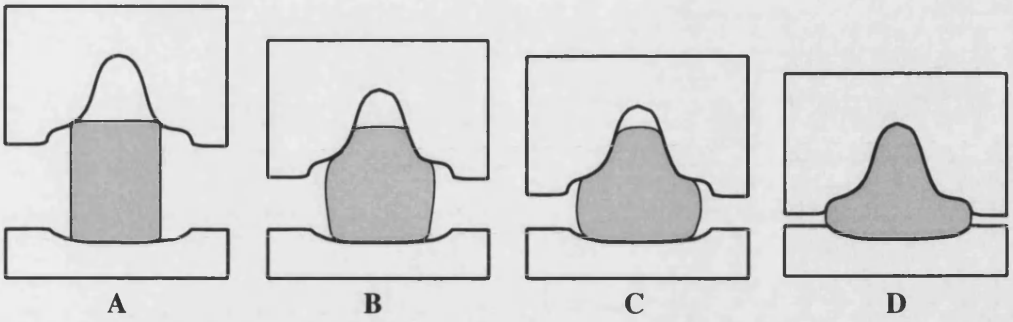
Furthermore, it is necessary to determine when and which contact region or regions will be released. Figure 6.6 shows that different reversed shapes result from releasing different contact regions.

An optimal contact condition for reducing the shape complexity may be achieved by searching for all possible combinations. However, this would be time consum-

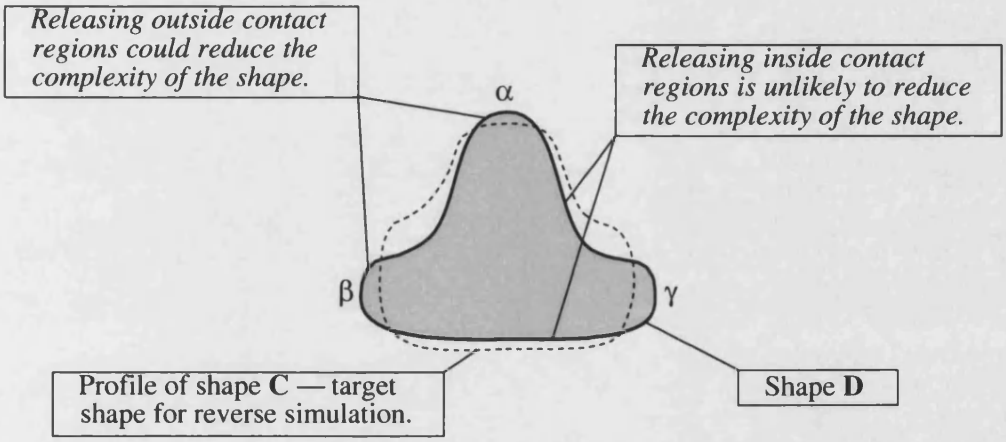
ing, especially for a complex shape using a finite element based simulation. For a relatively large time increment, several nodes may need to be released in order to reduce the complexity of the workpiece after a reverse step. By using a small time increment and releasing only one node per increment, the same nodes may be released within the same time. The exact time at which the node should be released is unknown and the releasing node procedure is not necessarily required for each small increment. This situation has been taken into account as described in later text. Due to the small time increment required in the simulation, the difference between the shapes resulting from the optimal combination for a big time increment and that following one-region-released condition in a small increment could be small and acceptable. Therefore, a more practical solution is to assume that only one region is released at each reverse increment. The identification of the region with the potential to reduce the complexity of the shape depends on the target shape.

The contact region furthest from the boundary of the target shape is regarded as the most complex region. This region should be released in order to reduce the shape complexity, i.e. the reversed shape will then become closer to the target shape. Moreover, in most forging processes, the bulk of material flows towards the cavities of dies. The amount of the contact region increases along the cavities and the material eventually reaches the deepest area. This contacting sequence is inverted in the reverse simulation, i.e. the contact regions are detached along the cavities from the deepest area. According to the above concepts, the contact region with the furthest distance to the boundary side of the target shape is released to form a new boundary condition and then to obtain a new reversed shape. Figure 6.7 shows target shapes identifying different contact regions with the furthest distance to the target profile in each case.

For the case in which all contact regions are inside the target shape, the releasing



(a) Forging process **A–B–C–D**.



(b) Comparison of two deforming shapes, **C** and **D**, shows that releasing outside contact regions of profile **C** in shape **D** could reduce the complexity after a reverse increment.

(c) Different locations of profile of shape **C** define different outside contact regions.

Figure 6.5: Target shape for reverse simulation.

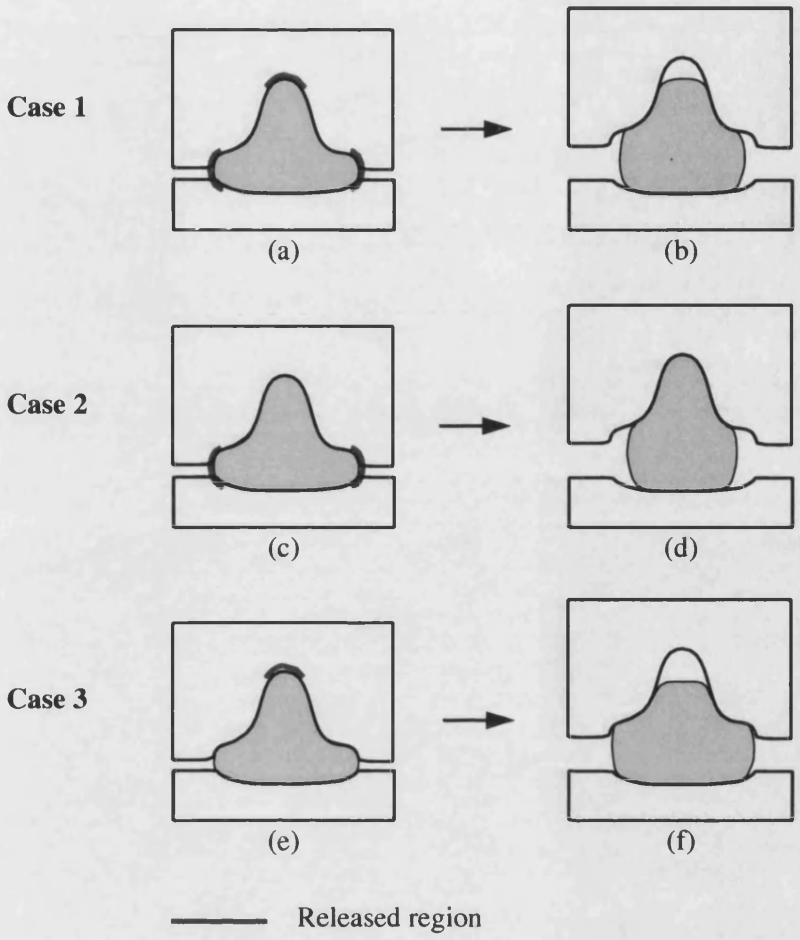


Figure 6.6: Different reversed shapes resulting from different contact conditions.

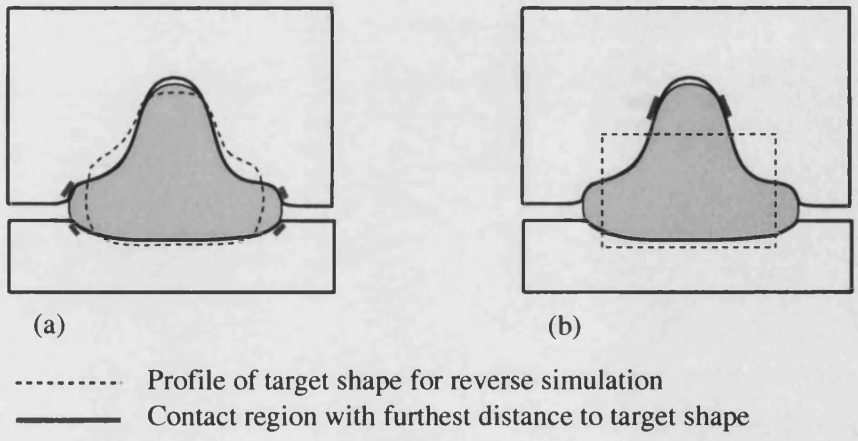


Figure 6.7: Furthest contact regions identified by different target shapes.

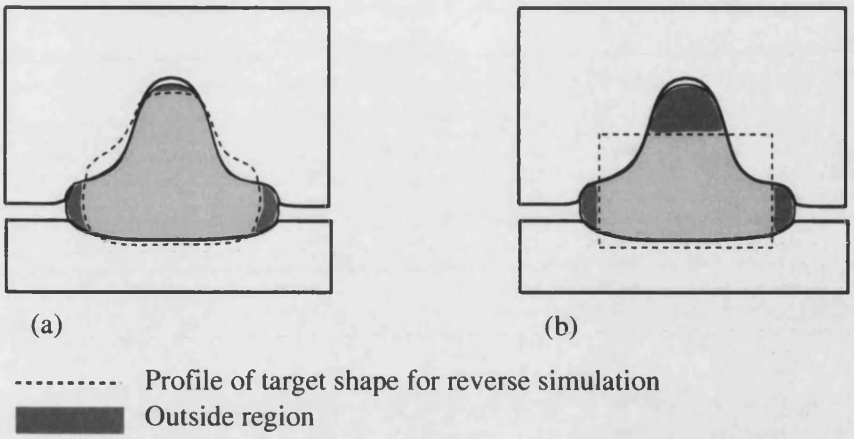


Figure 6.8: Complexity measured by the area of outside regions.

procedure is not carried out. If such an assumption is made, there are only two possible contact conditions, one-region-released and no-region-released. The no-region-released condition is taken into account since the time to release a region is unknown. A measure of the complexity of the shape is therefore needed and used to select a simple reversed shape for next increment.

The target shape is an ideal material distribution shape. The difference between the target shape and the deformed workpiece is one of the methods used to measure the complexity of the shape. It should be noted that two different boundary conditions result in two different shapes and thus the target shapes for the next increment are different in both cases. Each target shape is also different from the one in the previous increment. Figure 6.8 shows the area of outside regions used to calculate the complexity.

It is apparent from the above that to develop the reverse simulation based on the proposed concept of controlling boundary conditions, the following tasks need to be dealt with:

- To develop a method for defining the target shape — The development of

the method could be based on reducing the complexity of shape according to the material distribution.

- To locate the defined target shape — The method could be based on the minimum value of the complexity, i.e. the location provides the minimum difference between the target shape and a deforming (reversing) shape.
- To select two reversed shapes after two trial reverse increments — This requires a measure of the complexity of the shape.

If techniques to deal with the above tasks can be established, the target shape will be defined and thus the boundary conditions can be determined for the reverse simulation. The flowchart of a reverse increment is shown in Figure 6.9.

The proposed concept is for general use and should therefore be applicable to three dimensional forging problems of preform design. The following sections describe a method based on the second moment of area to achieve the requirement of using the proposed concept in reverse simulation for two dimensional forging problems.

6.4 Material Distribution

The material distribution of the workpiece varies during the forging process. In most cases, the initial shape of the workpiece needs to be a simple geometry, such as a rectangular or circular section for two dimensional plane strain applications. The reverse simulation should be able to find a deformation path, which always leads towards a simple distribution, from the finished product geometry to an initial billet. In order to satisfy this requirement, it is necessary to set a target

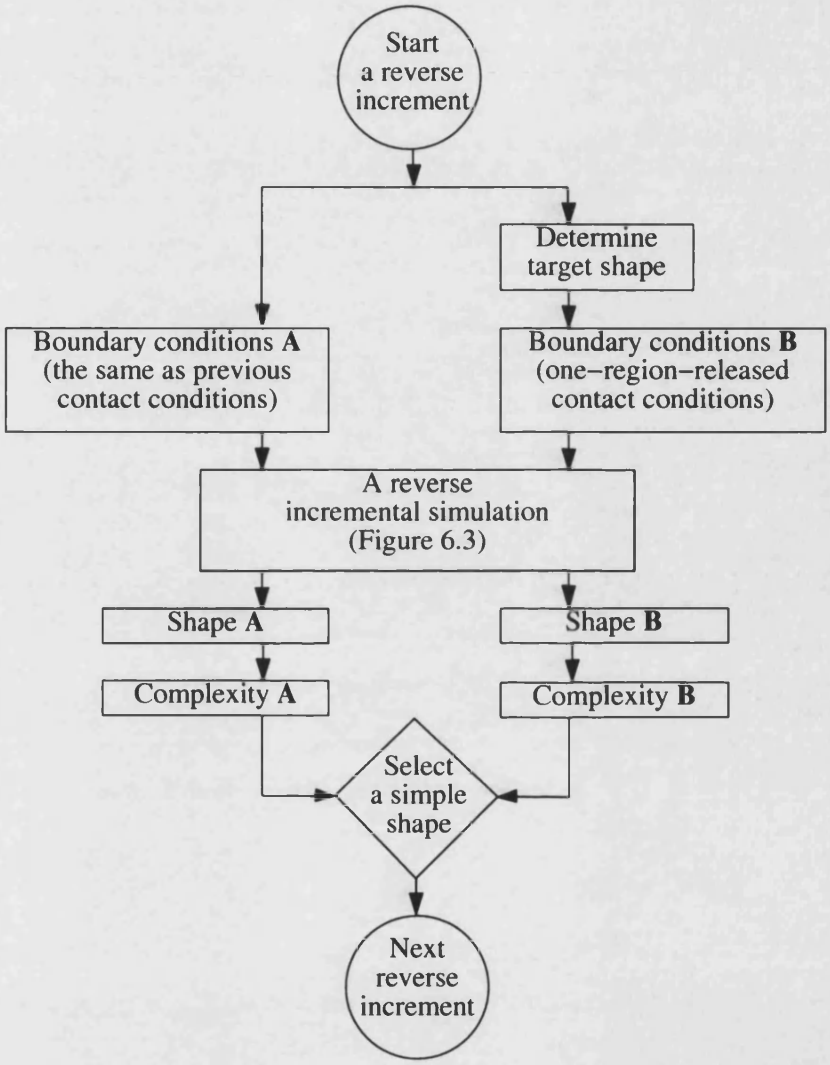


Figure 6.9: A reverse increment using the backwards tracing scheme with the control of boundary conditions.

shape with a simple material distribution to determine the boundary conditions for the reverse simulation. A rectangular section is supposed to be the billet section, i.e. the target shape, in this research for two dimensional plane strain applications.

In statistics, the variance

$$\sum \frac{(X-m)^2}{N} \tag{6.6}$$

describes a figure for the average variability of the scores, X, about the mean, m, expressed as squared deviations, where N is the number of scores. Due to the fact that the workpiece is composed of many particles, the deformation means that the position of the particles is changed. From a statistical view of the deformation of the workpiece, X is the position of the particle, m is the centre of gravity, and N is the number of particles. In order to evaluate the variance in a continuous domain, an integration form may be necessary. In this thesis, a similar concept based on the variance is proposed to evaluate the material distribution by calculating the second moment of area for plane strain cases .

Definition of the second moment of area

The second moment of area about the x-axis, I_x, and the y-axis, I_y, at the centre of the gravity (x_c, y_c) are defined as

$$I_x = \int (y-y_c)^2 dA \tag{6.7}$$

$$I_y = \int (x-x_c)^2 dA \tag{6.8}$$

where A is the bounded area. The value I_x and I_y of a shape can be obtained by means of Gaussian quadrature from a finite element model.

By comparing equation 6.6, 6.7 and 6.8, it can be noted that using the second moment of area to evaluate the distribution of the deformation is similar to

estimating the variance of the particle distribution. From a statistical point of view, the second moment of area about the x-axis, I_x, and the y-axis, I_y, measure how the material spreads over the orthogonal axes of the shape from its centroid. To satisfy the requirement of the "simplest" shape for the reverse simulation, it is necessary to reduce the variance of the particle distribution on the x-axis and y-axis at the same time during the simulation. Therefore, an appropriate combination of I_x and I_y could be used to search for the target shape, a rectangle, based on the consideration of the simple material distribution.

Rectangular Target Shape

The reverse simulation aims to an ideal rectangle with the second moment of area I_x^r and I_y^r about the x-axis and y-axis, respectively. To ensure the minimum difference of material spreading over both x-axis and y-axis between the ideal and deforming shapes, requires minimising the value D

$$D = \sqrt{(I_x^r - I_x)^2 + (I_y^r - I_y)^2} \tag{6.9}$$

For a rectangle, the value $I_x^r I_y^r$ is a function of area. That is

$$\begin{aligned} I_x^r I_y^r &= \frac{b^4 h^4}{144} = \frac{A^4}{144} \\ I_x^r &= \frac{bh^3}{12} \\ I_y^r &= \frac{b^3 h}{12} \end{aligned} \tag{6.10}$$

where h and b are the height and the width of a rectangle respectively, and $A = bh$ is the area. This equation constructs all possible ideal rectangles for the target shape with the same area as the deforming workpiece. By using equation (6.9) and (6.10), an ideal target shape can be defined based on reducing the second moment of area to represent the material distribution. In Figure 6.10, the point

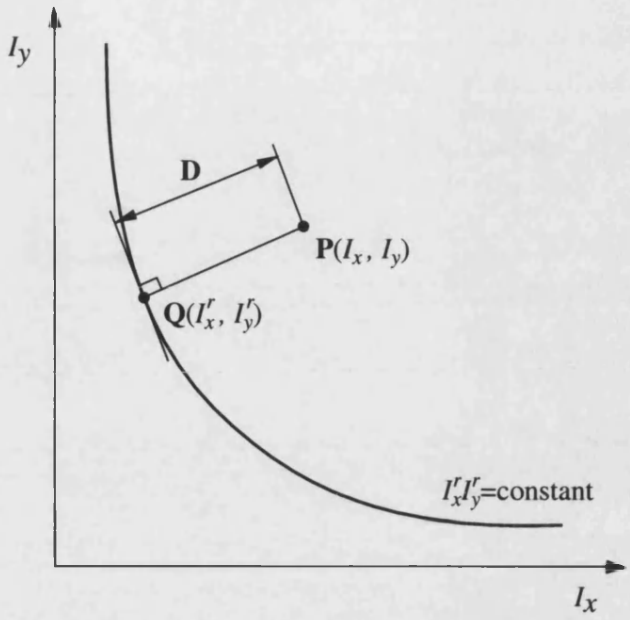


Figure 6.10: Curve $I_x^r - I_y^r$ for rectangular shapes with the same area A.

$P(I_x, I_y)$ representing the material distribution of the deformed workpiece has the closest point Q on the $I_x^r - I_y^r$ curve, i.e. equation 6.9 has the minimum value. The target shape is then defined by the I_x^r and I_y^r at point Q.

Location of Target Shape and Workpiece

To reduce the complexity of the shape requires a minimisation of the second moment of area. The minimum moment is at the centroid of the shape. Therefore, the centroid can be used to locate the workpiece and the target shape for specifying the outside contact regions.

Boundary Conditions and Complexity

In section 6.3, new boundary conditions were determined by releasing a contact area which is furthest from the target shape (see Figure 6.7). For a finite element model, the contact area is released by detaching the node from the die surface. Thus, two boundary conditions are obtained and two reversed shapes are predicted. The selection of these two predicted shapes for next increment requires an evaluation of the complexity of the shape. The complexity can be estimated

by the magnitude of the area outside the target shape as described in Figure 6.8(b). The shape with small area outside the target shape is then selected for next reverse increment.

The proposed method based on the second moment of area is used to set the target shape for controlling the material distribution. The method minimises the moment about the x-axis and y-axis and thus reduces the complexity of the workpiece during the reverse simulation.

6.5 Summary of Reverse Simulation

The proposed preform design approach uses the reverse simulation to predict a reversed path of the deformation. This approach employs a concept of the material distribution to determine a target shape which is a simple geometry, a rectangle for plane strain applications in this thesis. According to the target shape, new boundary conditions for a backward increment can thus be determined. Since the time to release the contact area is unknown, the no-area-released condition is also taken into account. Thus, two deformed shapes are obtained in each reverse increment. An estimate of the complexity is then used to select the simpler of the two shapes. By repeating the above procedure until all nodes are separated from the dies, a reverse shape with a simple material distribution can be predicted. If a non-rectangular reversed shape, i.e. a preform requirement, is found after the simulation, then another backward simulation needs to be carried out. The tool profile of the secondary reverse simulation is determined by the reversed shape which is found in the previous simulation. The procedure is repeated until a simple geometry is obtained.

The approach adopted for reverse simulation in preform design provides a convenient procedure for the control of the boundary conditions because using the distribution of material (the second moment of area for plane strain cases) as a control criterion depends only on the coordinates of the shape. The method looks for a reverse path which achieves the reduction of the complexity of the shape from the final product shape according to the selection of the material distribution close to a simple geometry at each increment. The approach finally finds an initial billet or, in cases in which preforming is required, a preform shape.

Chapter 7

Evaluation of Reverse Simulation for Preform Design

7.1 Introduction

Complete filling of die cavities is the main aim of the reverse simulation system, developed in this thesis. The system developed predicts preforms for plane strain forging processes. A comparison between predicted deformations and experimental results was therefore of major concern. Moreover, the forging load which provides a figure of the dissipation of energy was also investigated. The assessment was carried out using three experiments including one practical experiment for the one-stage forging process and two numerical experiments for multi-stage forging processes.

In the one-stage forging experiment, a billet predicted by reverse simulation from a finished product was forged in a practical experiment. The predicted billet was also used to run the forward simulation with a commercial package, FORGE2.

These results were analysed in order to understand the difference between the real process and the simulations including reverse and forward cases. The analysis was also used to evaluate the possibility of using the commercial software as a virtual forging press for doing experiments of the multi-stage forging instead of using a real forging machine.

In the multi-stage forging experiment, reverse simulations were carried out to predict preforms until a simple geometry, a billet, had been found. The forging machine was replaced by the commercial package FORGE2 to evaluate the preforms predicted by TEUBA.

7.2 Material Model for Commercial Package — FORGE2

FORGE2 is a commercial two-dimensional finite element package designed to simulate axisymmetric and plane strain forging processes. The package was developed by CEMEF (Centre de Mise en Forme des Materiaux) in France and is sold and supported by Transvalor S.A. [12]. FORGE2 uses the finite element analysis to model the mechanics of metal flow as well as thermal effects. The package provides a pre-processor for constructing the workpiece geometry and the die geometry, defining the process conditions, and outputting this information in the format required by the solver. A post-processor is supplied to visualise and analyse the result of the simulation. FORGE2 also incorporates automatic re-meshing which avoids excessive element distortion when analysing large deformations.

The description of the material behaviour in FORGE2 is based on the Norton-

Hoff law as follows,

$$s_{ij} = 2K(\sqrt{3}\dot{\bar{\epsilon}})^{q-1}\dot{\epsilon}_{ij} \tag{7.1}$$

where K is called "consistency", q is the sensitivity to the strain rate and $\dot{\bar{\epsilon}}$ is the effective strain rate. The consistency K depends on the thermo-mechanical conditions which are of various types. For example, the consistency can be defined by the strain-hardening power law combined with Arrhenius law for the temperature T as in the following formula:

$$K(T, \bar{\epsilon}) = K_0(\bar{\epsilon} + \epsilon_0)^n e^{\frac{\beta}{T}} \tag{7.2}$$

This formula requires the definition of the constant term K_0, the sensitivity to strain hardening n and the temperature term β. The effective strain $\bar{\epsilon}$ may tend to zero in the first increment of the calculation or in the non-deformed regions. This situation results in both the consistency K and the deviatoric stress s_{ij} tending to zero. The strain-hardening regulation ϵ_0 is therefore used to avoid this numerical problem when $\bar{\epsilon}$ tends to zero. The strain hardening regulation is chosen for numerical reasons only, and does not reflect the actual behaviour. The default value for ϵ_0 is 1×10^{-4} in FORGE2.

A material obeying the von Mises criterion is described as in the following form:

$$s_{ij} = \frac{\sigma_0}{\sqrt{3}} \frac{\dot{\epsilon}_{ij}}{\sqrt{\frac{1}{2}\dot{\epsilon}_{kl}\dot{\epsilon}_{kl}}} \tag{7.3}$$

or expressed in terms of the effective strain rate $\dot{\bar{\epsilon}}$ as

$$s_{ij} = \frac{2}{3}\sigma_0 \frac{\dot{\epsilon}_{ij}}{\dot{\bar{\epsilon}}} \tag{7.4}$$

The relation between the yield stress, σ_0, and the consistency, K, is then given

by

$$K = \frac{\sigma_0}{\sqrt{3}^{q+1}\dot{\bar{\epsilon}}^q} \tag{7.5}$$

Therefore, $q = 0$ can be considered as a limiting case of Norton-Hoff law for a material obeying the von Mises criterion with a yield stress $\sigma_0 = \sqrt{3}K$.

In this thesis, an aluminium alloy BS6063F is selected for the experiments and simulations. The flow stress $\overline{\sigma}$ can be described by the strain-hardening law with the material strength C and the strain hardening n as

$$\overline{\sigma} = C\overline{\epsilon}^n \tag{7.6}$$

For simulations with isothermal conditions, the β is set to zero. The consistency K is therefore expressed as

$$K = K_0(\overline{\epsilon} + \epsilon_0)^n = \frac{C}{\sqrt{3}}(\overline{\epsilon} + \epsilon_0)^n \tag{7.7}$$

FORGE2 also models friction on the contact interface between the workpiece and die surfaces. The Tresca friction law, as described in Chapter 3 (Equation 3.37) and also used in the modelling of the reverse simulation, is selected for the forward simulations of FORGE2 to reflect the friction effects in forging process.

7.3 Case Study 1: One-cavity Forging

7.3.1 Introduction

A desired forging product with simple geometry could be made by one-stage forging with appropriate process conditions if the size of the billet is carefully selected or properly increased. It is also sometimes possible to forge products with complex geometry in one stage by increasing the volume of the billet without the consideration of the wastage of the material and the energy, the maximum forging load and the quality. This case study concentrates on predicting the billet for a one-stage forging and understanding differences between the theoretical reverse and forward simulations and the practical work. The experiment may be regarded as a forging produced with a minimal flash by determining a suitable billet size to forge the product although in practice flash may be required to permit variations in billet volume.

A rigid-perfectly plastic model with a constant flow stress is valid for hot forging. Due to the complexity of controlling the temperature, the experiments were performed using a soft material, aluminium alloy BS 6063F, at room temperature. The simple compression test was used to obtain the flow stress for the rigid-perfectly plastic and the strain-hardening model. Moreover, ring compression tests were also carried out to determine the friction factor.

According to the designed profiles of both product and tools, the finite element mesh models were constructed using the geometric modeller, CADfix [112], and were exported into TEUBA to run the reverse simulation for predicting a billet with a rectangular section. The section was then used to do the practical experiment. A numerical forward simulation using a commercial software FORGE2

was also performed for this case. Finally, the results of the simulations were compared with the experiments in terms of deformation trend and forging load.

7.3.2 Forging Press and Data Acquisition System

A CNC hydraulic forging press and a data acquisition system were used for the experiments. The press is capable of a forging force of 100 tonnes and a constant ram speed of 21 mm per second [119]. A load cell was fixed between the ram and the top die, and connected to a computer with an interface to record the forging load. The displacement of the top die was measured by a displacement transducer during the process. The main components of the forging press and the recording device are illustrated in Figure 7.1. Figure 7.2 shows the hydraulic forging press and the computer which were used to carry out the experiments in this research work.

7.3.3 Flow Stress — Simple Compression Test

Simple compression tests were carried out to determine flow stress under a constant tool velocity of 21 mm per second at temperature around 17 °C for aluminium alloy BS 6063F. The specimens were made with a diameter to height ratio of 1.5 according to the recommendation of ASTM [120]. A typical specimen is shown in Figure 7.3(a). To reduce the barrelling effect of the specimen profile caused by friction, a bearing grease lubricant, ROCOL MTS 1000 [121], was applied on the contact surface. The experimental results in Figure 7.3(c) show that the barrelling effect was not significant up to a true strain of 1.26.

Neglecting friction, the true compressive stress σ required to producing yielding

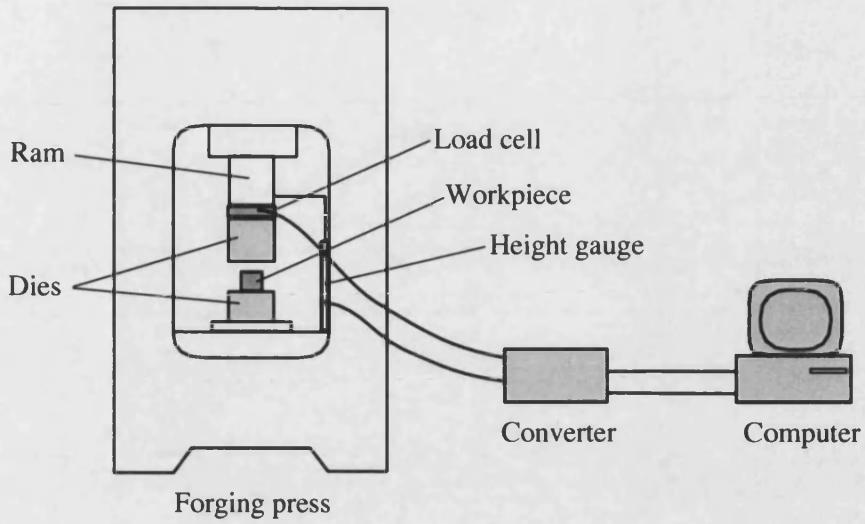


Figure 7.1: Main components of forging press and recording device.

Figure 7.2: Hydraulic forging press and computer for recording data.

during the compression tests is

$$\sigma = \frac{F_c}{A} = \frac{4F_c}{\pi D^2} \tag{7.8}$$

where F_c is the uniaxial compressive force, A is the area of cylinder section, D is the diameter of the cylinder. According to the law of constancy of volume,

$$D_0^2 h_0 = D^2 h \tag{7.9}$$

where D_0 and h_0 are the initial diameter and height of the cylinder respectively. The compressive stress is

$$\sigma = \frac{4F_c h}{\pi D_0^2 h_0} \tag{7.10}$$

The experimental data of flow stress against the true strain is plotted in Figure 7.4, in which the true strain ϵ is given by

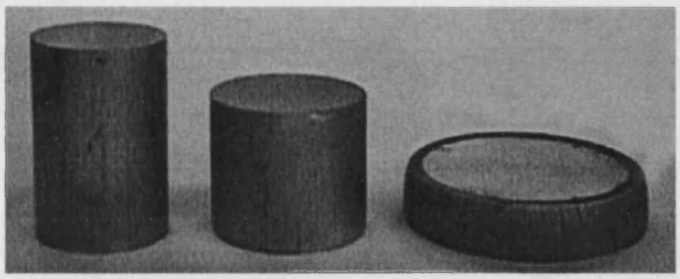
$$\epsilon = ln\frac{h_0}{h} \tag{7.11}$$

To model the flow stress curve, a data analysis package SPSS [122] was employed. For the rigid-perfectly plastic material model, the yield stress was $Y = 176.92$ MPa, and the strain-hardening model was $\overline{\sigma} = 194.33\overline{\epsilon}^{0.183}$ MPa. The results are shown in Figure 7.4, and were used to run the reverse and forward simulations.

7.3.4 Friction Factor — Ring Compression Test

The ring compression test, developed by Male and Cockcroft [117], is probably the most commonly used method for determining friction factors in metal forming. The test involves the compression of a hollow ring, and the determination of the variation of the internal diameter with the height reduction. The method involves measuring the variation of diameter with height which depends on the friction between the ring and the die surface, and provides a sensitive response to friction effects [118]. When using the ring compression test to determine the friction

factor for a particular set of experimental data, a set of calibration curves are required. The curves determined by Hawkyard and Johnson [118] assume that the material follows a rigid-perfectly plastic model without barrelling so that the deformation is completely homogeneous with uniform states of stress and strain. These curves are used in this thesis as the calibration curves for the aluminium specimens.



(a) Specimen size = ϕ25x37.5 mm. (b) True strain = 0.38 (c) True strain = 1.26

Figure 7.3: Specimen and experimental results from the simple compression test.

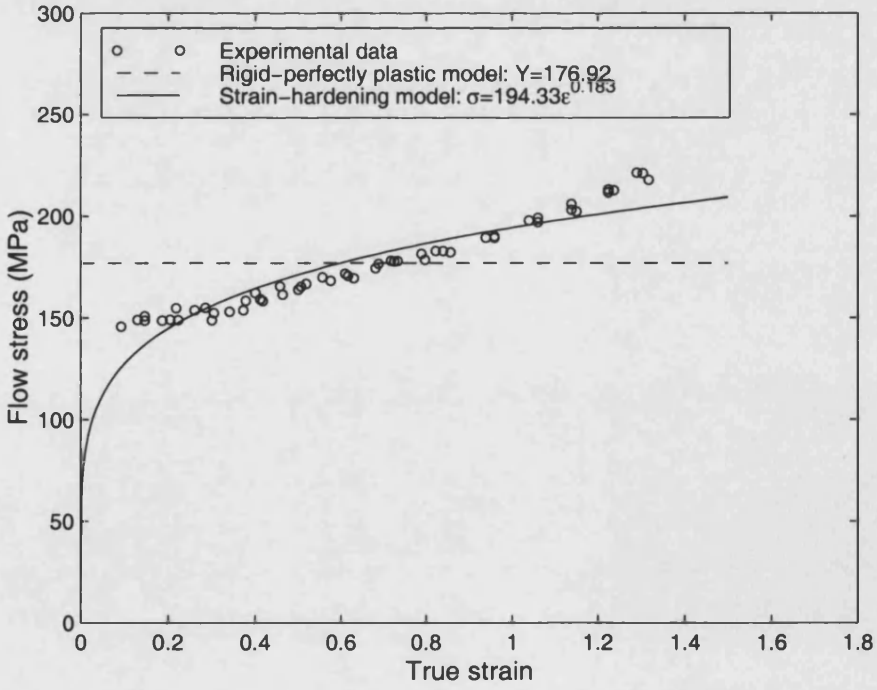
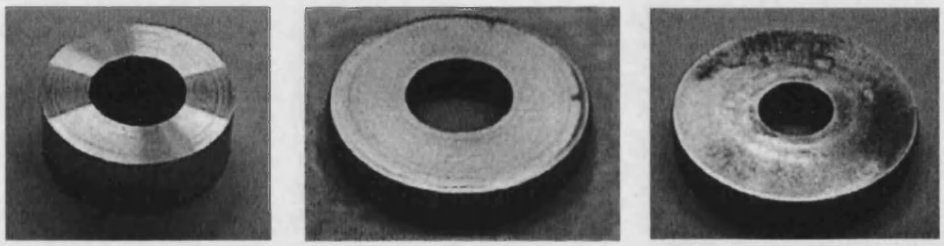


Figure 7.4: Experimental data and predicted flow stress curves for aluminium alloy BS 6063F.

The ring specimens were made of aluminium alloy BS6063F in ratio 6:3:2 of outer diameter, internal diameter and thickness respectively. Three tests with five incremental steps for each condition, dry or without lubrication, were carried out up to about 50 per cent of reduction in height at room temperature (The actual temperature was around 15 °C). In the cases with lubrication, applied lubricant is graphite-based bearing grease made by ROCOL company [121].

Figure 7.5 shows the original specimen and the deformed rings. Friction factors are then obtained by the interpolation of the recorded data into the calibration curves shown in Figure 7.6. The factor is determined as 0.725 for the case without lubrication, and 0.085 for the case of applied lubricant.



(a) Specimen Size (ID:OD:H) =12.5: 25: 8.33 mm.

(b) 53 % reduction with lubrication.

(c) 50 % reduction without lubrication.

Figure 7.5: Specimen and experimental results from the ring compression test.

7.3.5 Finished Product and Tools

The finished product shown in Figure 7.7 was designed for both experiments and simulations. The designed radii, draft angles and fillets were determined by the available sizes and shapes of milling cutters in order to easily machine the profiles of tools. The selected material for the product was aluminium alloy BS 6063F. According to the profile of the finished product, tools for the experiment were designed as shown in Figure 7.8. The dimensions of dies are given in Figures

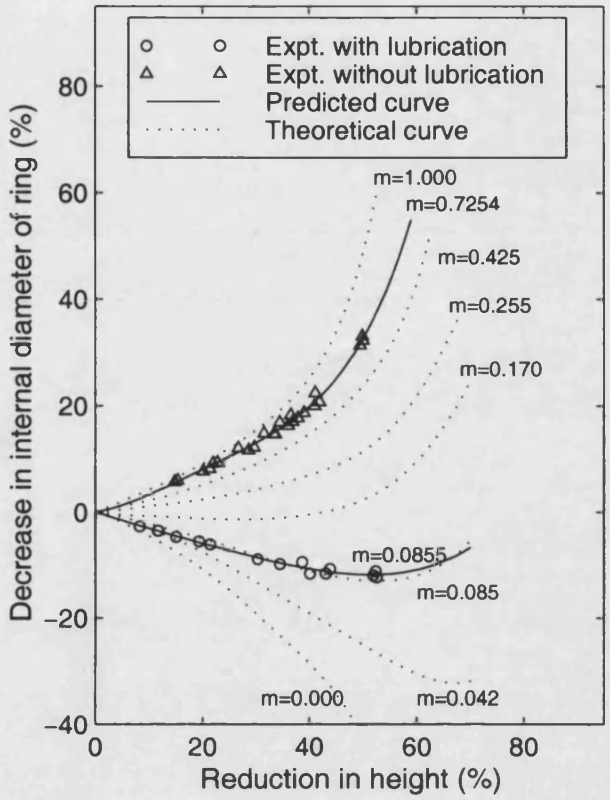


Figure 7.6: Measured friction factors and theoretical curves after Hawkyard and Johnson [118]

A.1, A.2 and A.3 in appendix. The flash land and the gutter were treated as a straight line for both bottom and top die in view of reducing the cost of machining. The die material was AISI H13 Steel, a 5% chromium ultrahigh-strength steel [123]. The dies were also heat treated and ground to provide appropriate material properties and surface finish.

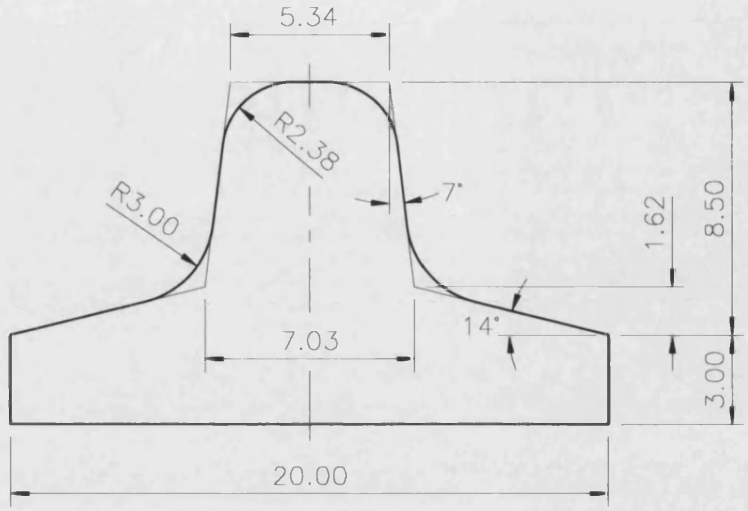


Figure 7.7: Finished product for the one-stage forging experiment.

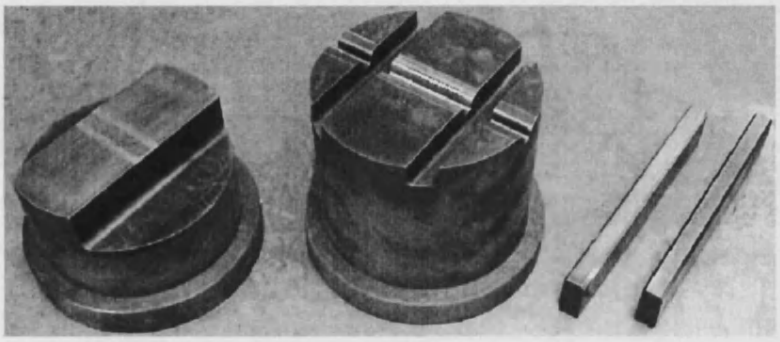


Figure 7.8: Forging dies for experiments.

7.3.6 Material Models and Process Conditions for Simulations

Material models and process conditions for both reverse and forward simulations are summarised in Table 7.1. For the reverse simulation using TEUBA, the magnitude of the die velocity does not affect the upper bound solution of forging load. The die velocity was set as a constant value of 1 mm per second. In the forward simulation using FORGE2, a die velocity of 21 mm per second corresponding to the ram speed of the press was used.

Table 7.1: Material models and process conditions for the reverse and forward simulations of Case Study 1

Simulation type	Workpiece material	Flow stress (MPa)	Tresca friction factor	Die velocity (mm/sec)
Reverse	Rigid-perfectly plastic	176.92	0.085	1
Forward	Strain-hardening	$194.33\bar{\epsilon}^{0.183}$	0.085	-21

7.3.7 Deformation Trend

The reverse simulation starts from the finished shape as shown in 7.10(a). As the top die moves upwards, the material gradually flows into the target shape which is determined by the material distribution measured by the second moment of area. During the reverse deformation, the target shape varied according to the reversed shape. The complexity of the workpiece shown in Figure 7.9 is being reduced, i.e. the reversed shape progresses towards a simple geometry. Finally, the simulation predicted a shape close to a rectangle as shown in Figure 7.10(f). This was then regarded as the billet for this one-stage forging. The size of the predicted billet was 10.66 mm by 11.69 mm.

The predicted billet in Figure 7.10(f) was used to run a forward simulation by using commercial package FORGE2. The trend of the deformation as shown in Figure 7.11 is generally similar to that predicted by the reverse simulation. Nevertheless, there are two features of Figure 7.11(f) that are different from Figure 7.10(a): (1) a small unfilled area occurred in the cavity; (2) a different flash shape formed in the lower right corner.

Figure 7.12 shows experimental results from the billet to the finished shape. In comparison with the experimental results shown in Figures 7.13, 7.14 and 7.15, the predicted profiles from the forward simulation are generally similar. For those

predicted by the reverse simulation, the deformation located in the bottom-right areas is different from the experimental results. The predicted velocity field, which represents the minimum energy dissipation for the deformation, was based on the rigid-perfectly plastic model. This approximation of no strain hardening would be expected to have most effect where strain is high such as the bottom right-hand region in Figure 7.13. By comparing with reverse simulation, the profile predicted by the forward simulation using the strain-hardening model is closer to the experimental results, especially in Figures 7.14 and 7.15 in which strain variation is higher in the bottom-right corner. Moreover, different surface finish may exist in the different dies between the ring compression test and this experiment. The measured friction factor may be slightly lower than that of this experiment because simulations predicted more deformation in the bottom-right location. Furthermore, the forward simulation using six-noded triangle elements may predict a more flexible profile than that predicted by reverse simulation using three-noded triangle elements. The discussion of the effect of mesh density and

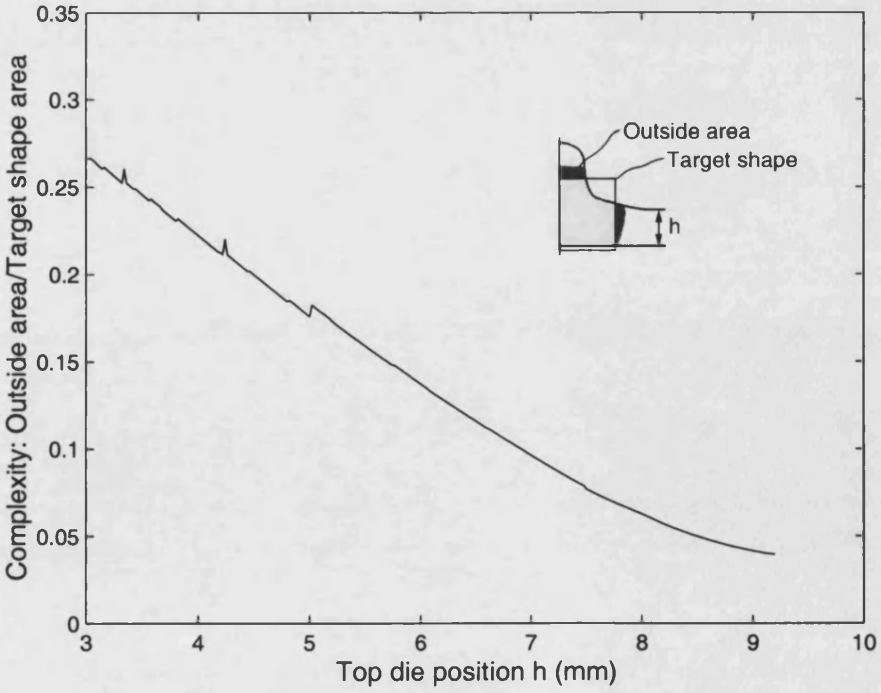


Figure 7.9: Complexity of workpiece during the reverse simulation of Case Study 1.

element type is in section 7.3.10.

7.3.8 Velocity Field

The velocity field is able to show whether material flows are in balance which is important for reducing forging load and producing good quality forgings. The velocity field also provides the information about where the material flow, in contact with die surfaces, is at a high speed, and hence indicates areas in which potentially high die wear could occur.

The velocity field, Figure 7.16, predicted by the the reverse simulation, shows that material flows reversing from the bottom right corner are faster than those from the top cavity. The die wear on the surface of the top die may be more serious than that of the bottom die because of a higher velocity field on the contact surface. Moreover, Figure 7.16(f) corresponding to an early stage of deformation has a small velocity at the bottom left corner, indicating that a dead zone could exist during the early stage of deformation. These results are generally in accord with those predicted by the forward simulation as shown in Figure 7.17 although the scale of velocity is different. The predicted velocity fields in both reverse and forward simulations are also similar to those of the experimental results which can be seen from macrostructure photographs shown in Figures 7.13, 7.14 and 7.15 although the reverse simulation predicts different velocity distributions with a higher value in the bottom-right areas.

7.3.9 Forging Load

Forging loads predicted by simulations and acquired from experiments are represented by the corresponding flash height as shown in Figure 7.18. The experimental curve was obtained from three incremental forging processes because of the need to apply lubricant.

The experimental error on the displacement measurement could be caused by the elasticity of the press. By measuring near the centre of the press bed and the position of the height gauge, the displacement difference at the load of 37 tonnes, which is about 10 tonnes higher than the maximum load in this experiment, was recorded at 0.145 mm. The maximum error of the height gauge was in a range of 0.147 mm according to the variation of the recorded data. Therefore, the experimental error caused by the elastic displacement of the press and the variation of the height gauge is 0.292 mm. The effect of this error may affect the measurement of flash height, and thus the recorded load curve. The curve shifted by this error is shown in Figure 7.19.

The maximum response error of the load cell was estimated as 1.528 tonnes after repeating the calibration procedure. This value corresponding to this plane strain experiment was 0.299 kN/mm which was divided by the length of the specimen of 50 mm. In comparison with the maximum recorded forging load, the error is about 6 per cent.

The forging load increases as the flash height decreases. This phenomenon results from the increase of the required energy for deformation which can also be explained by Figures 7.16 and 7.17 in which the velocity of material flows is higher at the end of the process. The forging load predicted by the reverse simulation is higher than that predicted by the forward simulation and the experiments. This

can be expected when using the upper bound theory. Furthermore, the forward simulation also predicts higher forging loads except at the early stage of deformation. The difference in forging loads between the simulations and the experiments could result from the approximate modelling of forging process, especially from the modelling of plastic deformation since the friction effect is small in this case. The latter is illustrated in Figure 7.20 which shows that most of the forging load is caused by plastic deformation and not by friction.

7.3.10 Discussion of Effect of Mesh Density and Element Type

The effect of mesh density and element type was investigated in terms of different construction of meshes executed in reverse and forward simulations. The number of elements and nodes, and the computational time which is discussed in the next section, are listed in Table 7.2.

Table 7.2: Construction of meshes and computational time of Case Study 1

Simulation type	Element type	Case number	Number of nodes	Number of elements	CPU time (seconds)
Reverse	three-noded triangle	1R1	310	539	551
		1R2†	464	826	1434
		1R3	751	1396	5696
Forward	six-noded triangle	1F1	285	122	3208
		1F2†	351	154	9212
		1F3	967	444	25067

†The cases were compared with the practical experiments.

In reverse simulation using three-noded elements, the deformation trend shown in Figure 7.21 is similar in the three cases using different mesh densities. However, some local regions in the final results, Figures 7.21(c), (f) and (i) are slightly different. By increasing the number of elements, more material remained in the

bottom-right corner and the top cavity of the predicted preforms. A further investigation for complete die fill was carried out using a forward simulation which started from the predicted preform of Figure 7.21(i). The simulation result, Figure 7.22, shows that a complete die fill can be achieved. Potential folds may exist in two regions with concave curves. This could be caused by the jagged boundaries in the predicted preform. However, the shape of flash was close to a straight line which was demanded in the original design of the final product shape. This result shows that reverse simulation has the potential in industrial preform design to achieve a complete die fill. The boundary of the workpiece predicted by the forward simulation using six-noded elements is smoother and more flexible, as shown in Figure 7.23. Moreover, the difference of deformation trend between cases is very small. Therefore, the effect of mesh density on deformation in the forward simulation is smaller than that in reverse simulation.

The velocity fields, shown in Figure 7.24 predicted by reverse simulation, are generally similar in the early reverse stage. As the flash gap increases, the mesh density has more influence on the predicted velocity fields, especially in Figure 7.24(c) which used coarser meshes. However, the effect of mesh density on the velocity fields has less influence on the forward simulations shown in Figure 7.25. The same situation was also reflected on the forging load that the results predicted by the six-noded element model, shown in Figure 7.27, were not affected by the mesh density and was better than those predicted by the three-noded element model, shown in Figure 7.26.

In terms of the comparison of the deformation trends, the velocity fields and the forging loads calculated by using different mesh densities, it can be concluded that the six-noded triangle element may provide a better prediction than the three-noded triangle element. However, the simulation using six-noded elements may require more computational time which is discussed in the next section.

7.3.11 Computational Time

The type of elements and the number of nodes and elements are different between the reverse simulations and the forward simulations. This causes difficulties in comparing the central processing unit (CPU) time to run the simulations. Other factors, such as the re-meshing procedure, will also affect the computational speed. Therefore, this section provides an approximation of CPU time in both reverse and forward simulations. The re-meshing procedure in the reverse simulation was carried out using a geometric modeller, CADfix [112], and the time for performing the procedure was not taken into account. Table 7.2 shows that the reverse simulation is faster than the forward simulation. One of the reasons could be the fact that the numerical integration is not needed by the three-noded element formulation used in the reverse simulation while the integration procedure is performed in each iterative calculation of the forward simulation using six-noded elements. Moreover, the reverse simulation directly predicts a preform, a billet in this one-stage forging case, that might otherwise need to be obtained by multiple trial-and-error runs using the forward simulation. This could be a significant gain of reducing lead-time by using the proposed method, i.e. the reverse simulation approach, for preform design in the forging industry.

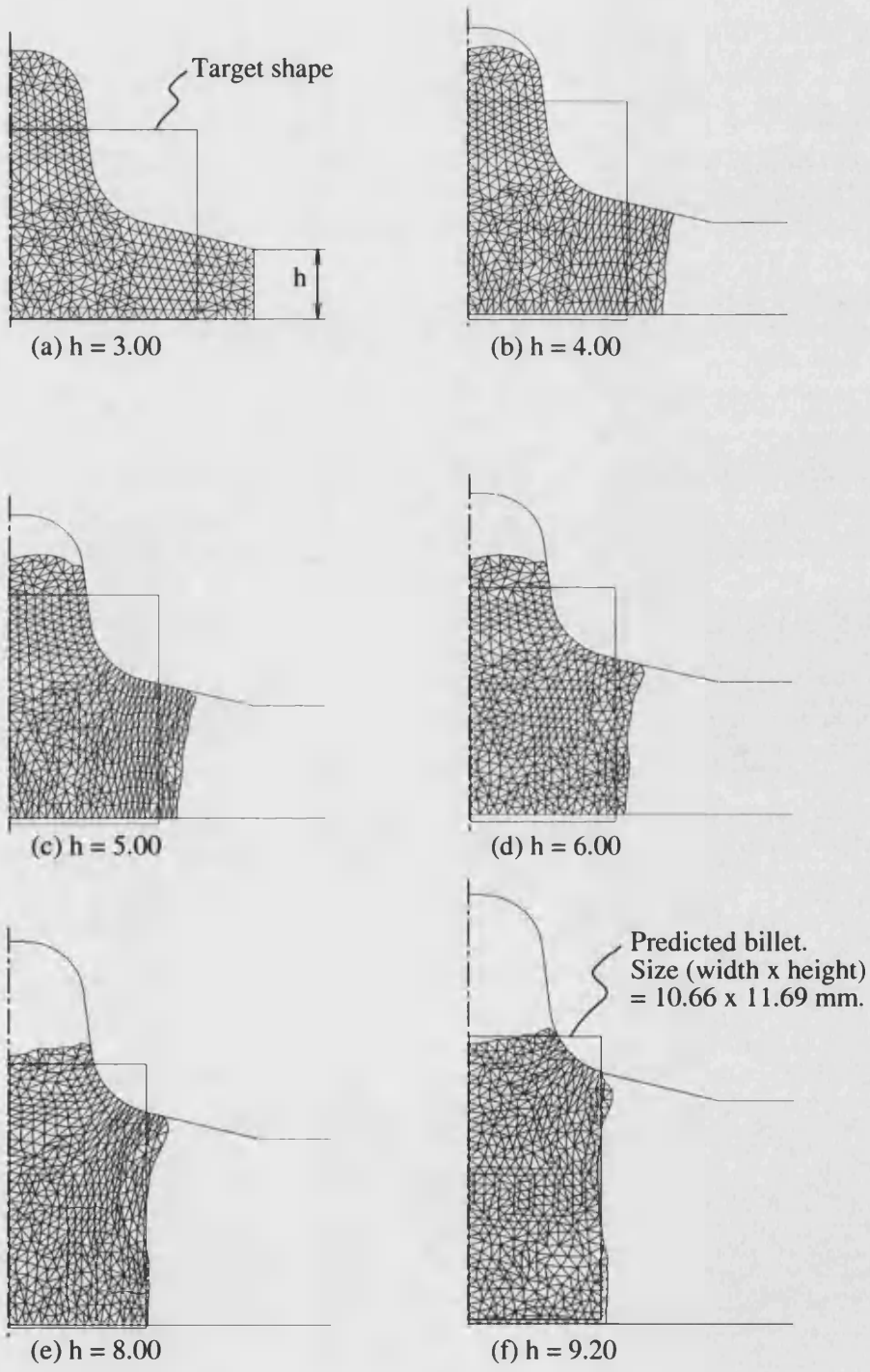


Figure 7.10: Reverse simulation and target shapes of Case Study 1.

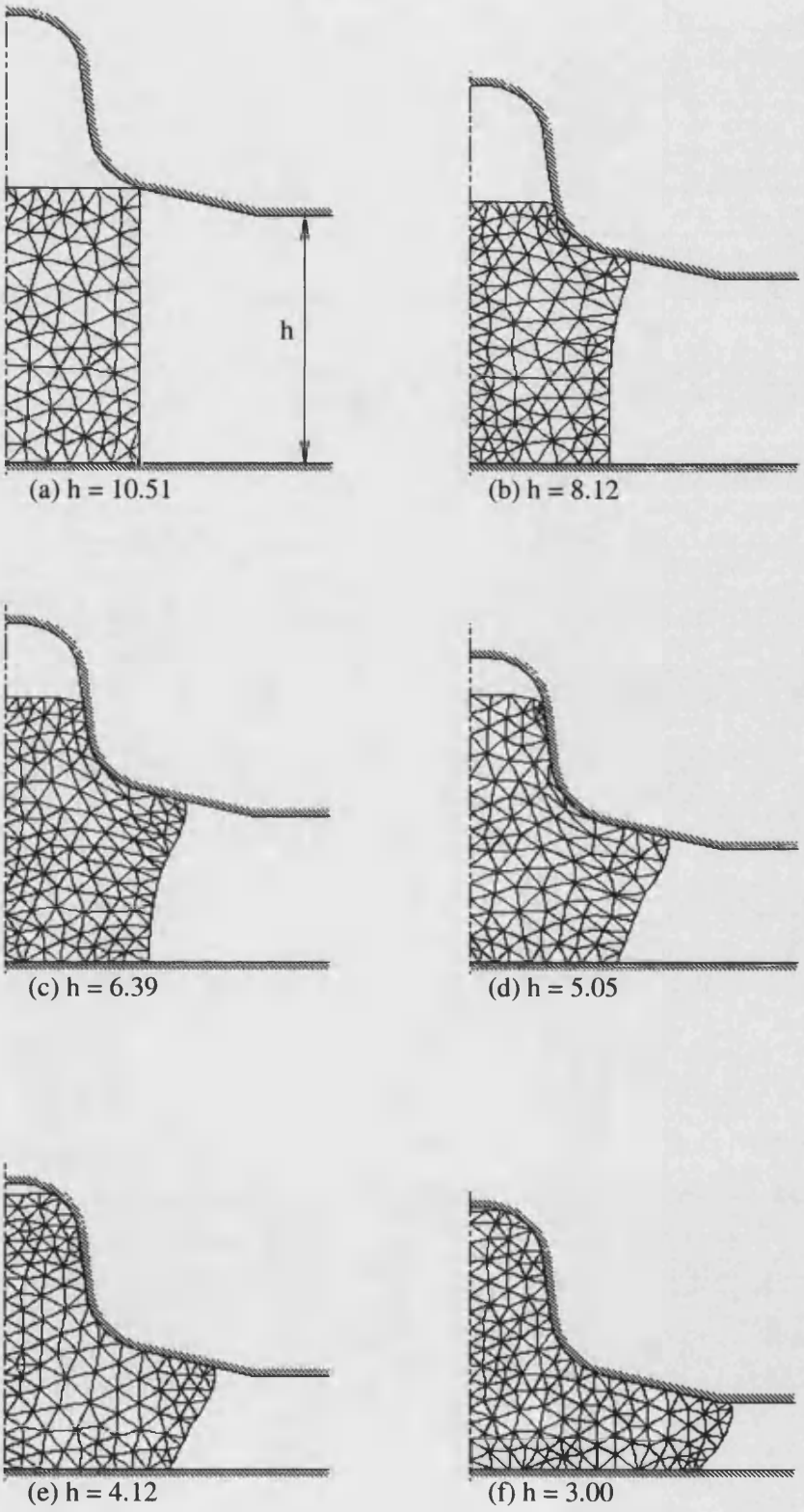
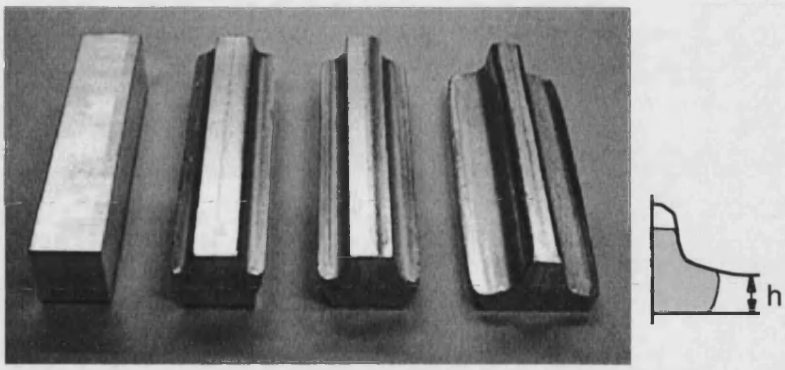


Figure 7.11: Forward simulation of Case Study 1.



(a) h= 11.69 (b) h= 6.87 (c) h= 4.60 (d) h= 3.05

Billet size(w x h) = 10.66 x 11.69 mm.

Figure 7.12: Experimental results of Case Study 1.

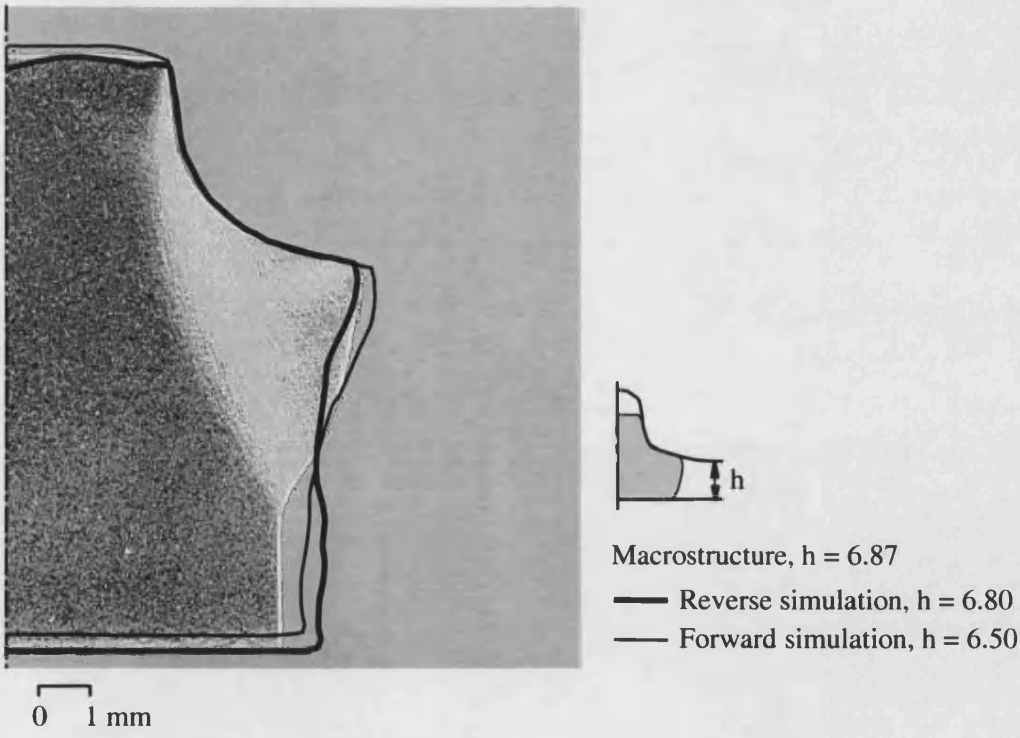


Figure 7.13: Macrostructure and predicted profiles from the reverse and forward simulations of Case Study 1 — (I).

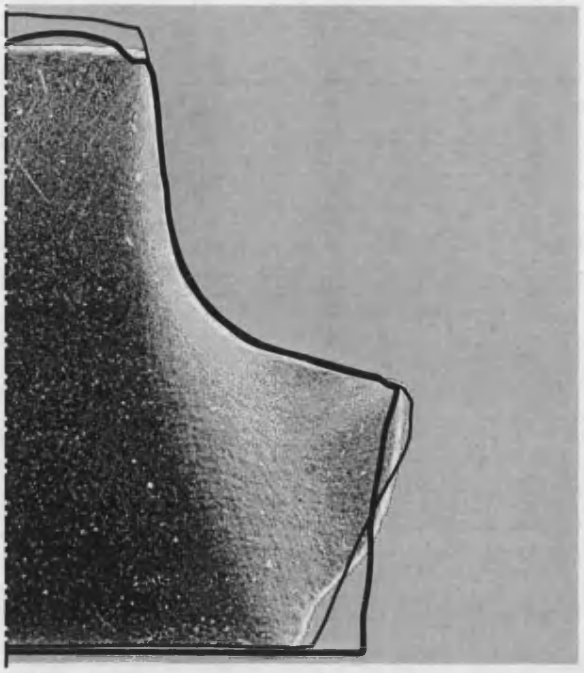


Figure 7.14: Macrostructure and predicted profiles from the reverse and forward simulations of Case Study 1 — (II).

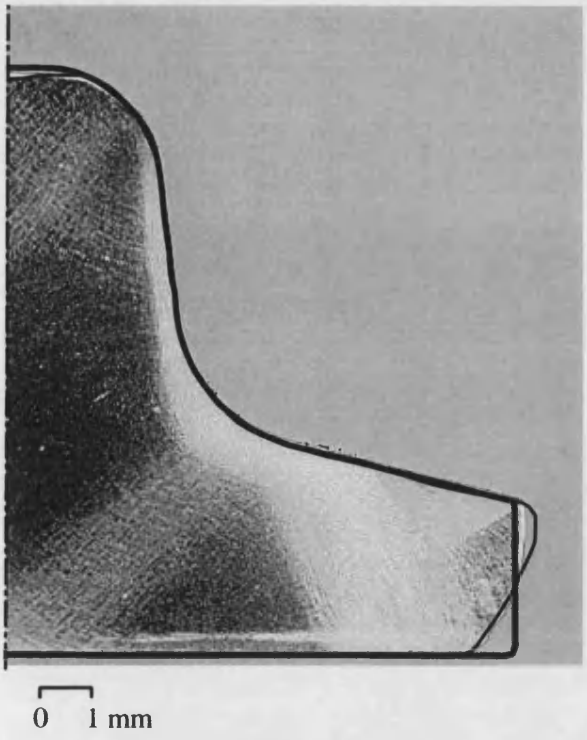


Figure 7.15: Macrostructure and predicted profiles from the reverse and forward simulations of Case Study 1 — (III).

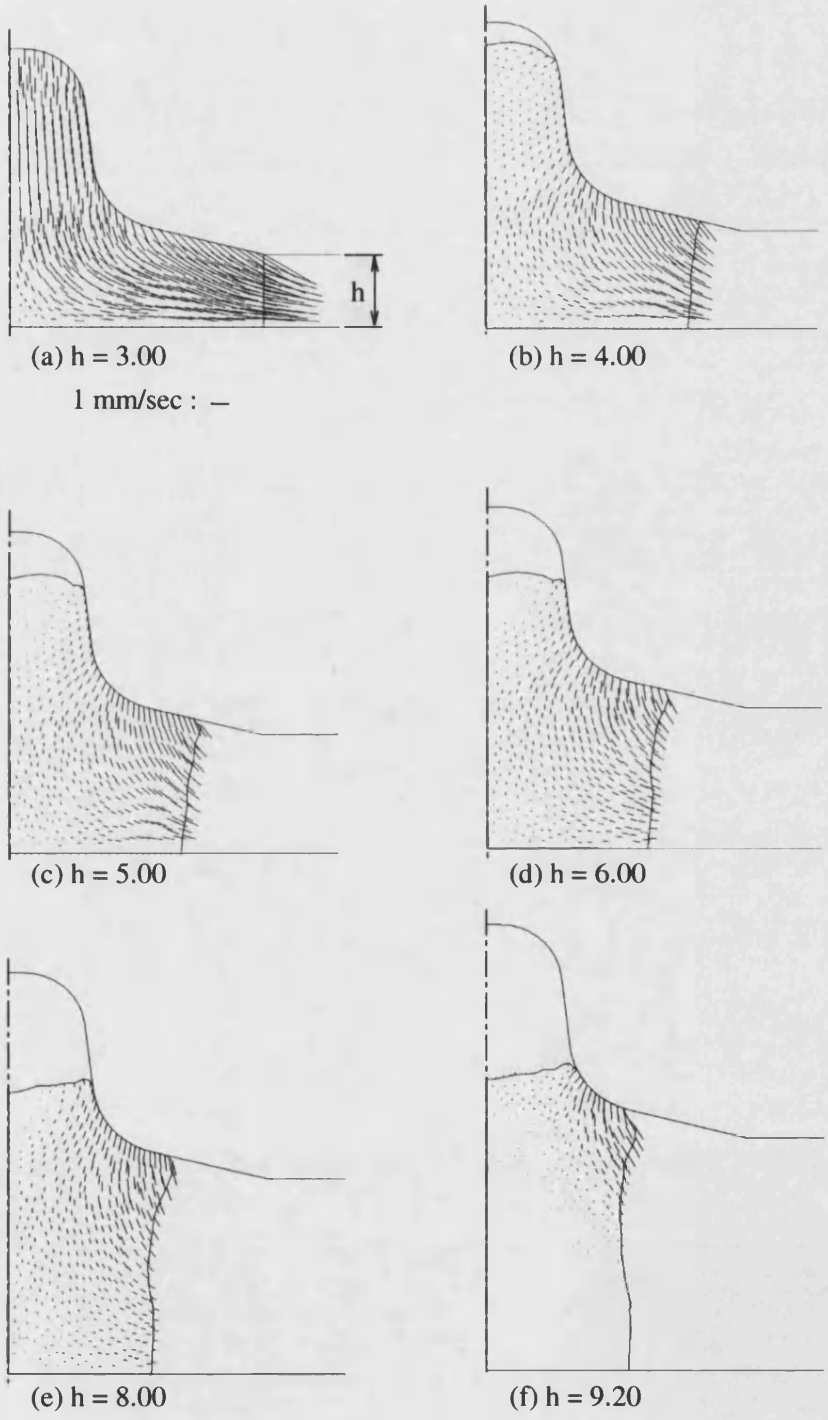


Figure 7.16: Velocity fields in the reverse simulation of Case Study 1.

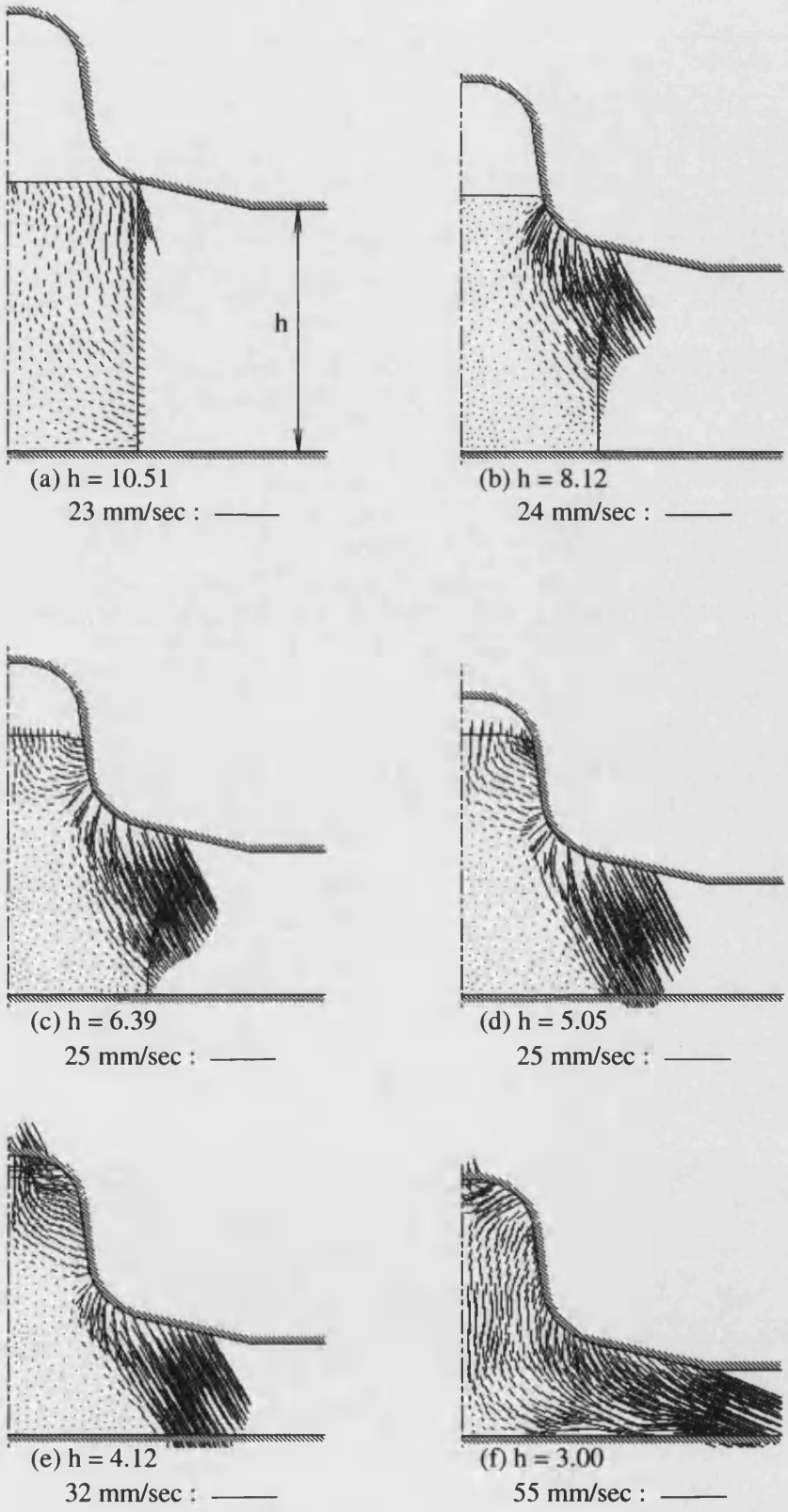


Figure 7.17: Velocity fields in the forward simulation of Case Study 1.

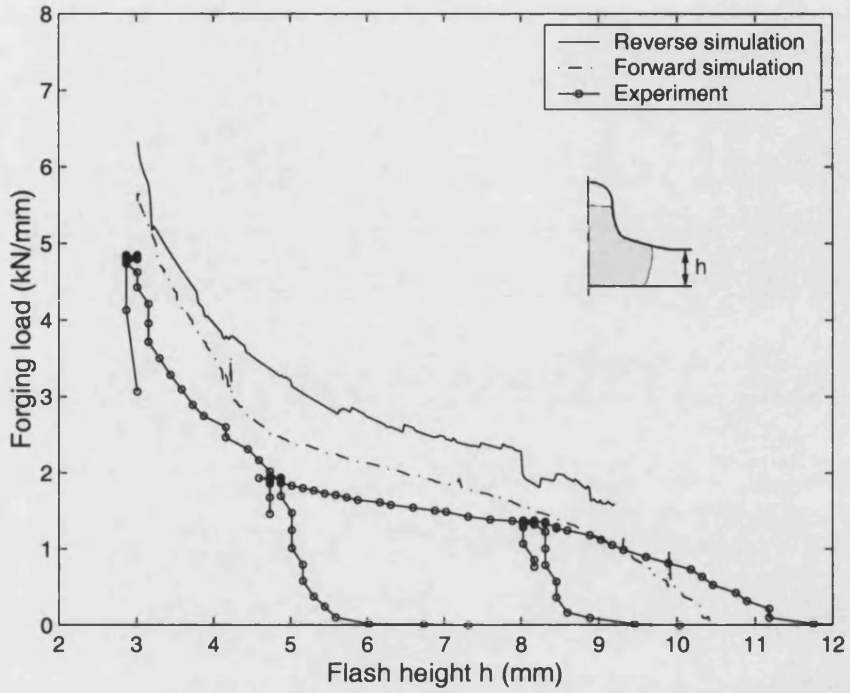


Figure 7.18: Load comparison between the simulations and experiment for Case Study 1.

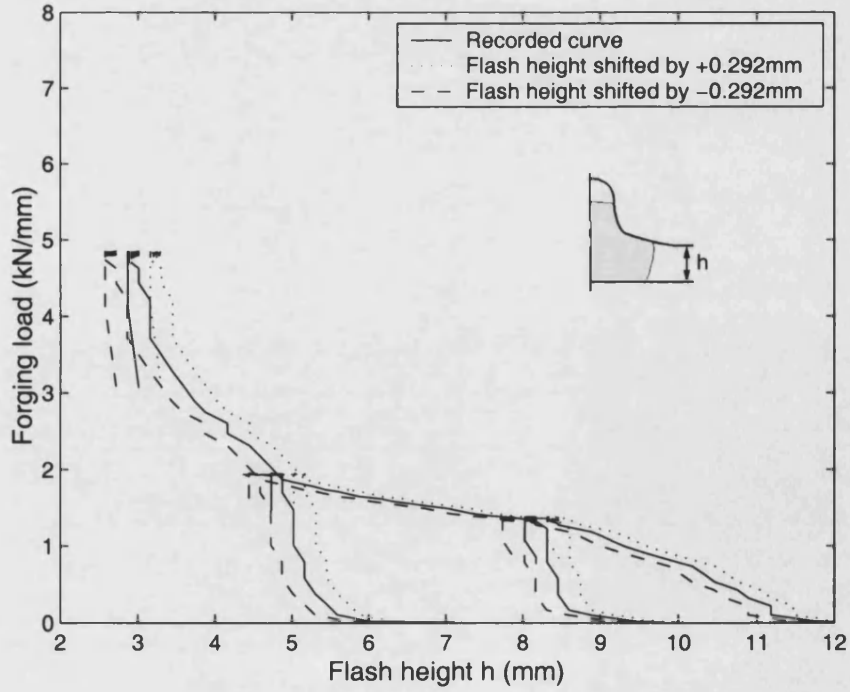


Figure 7.19: Load variation caused by possible measurement error of flash height.

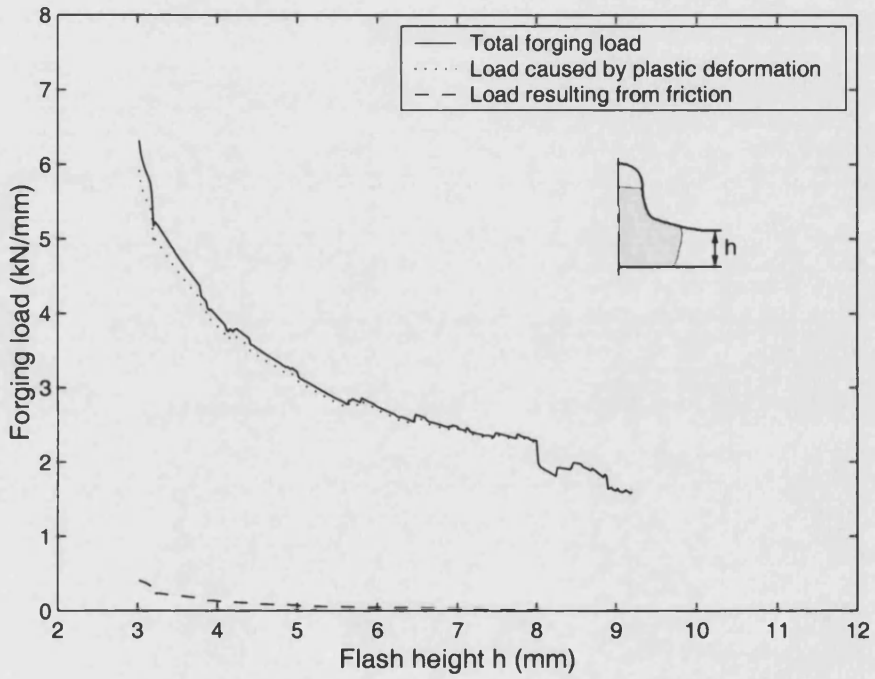


Figure 7.20: Load predictions for the reverse simulation of Case Study 1.

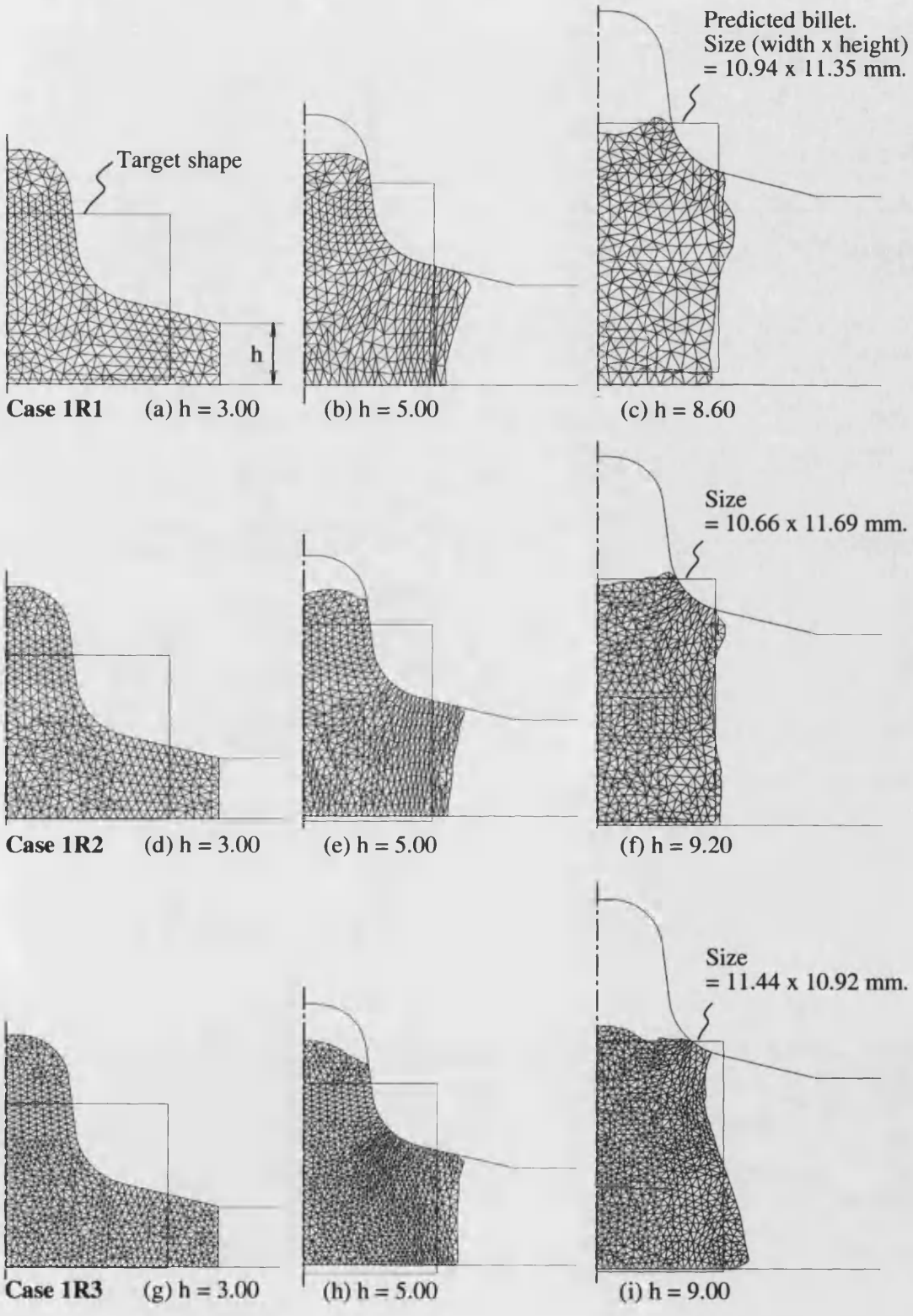


Figure 7.21: Reverse simulations and target shapes resulting from different mesh densities in Case Study 1.

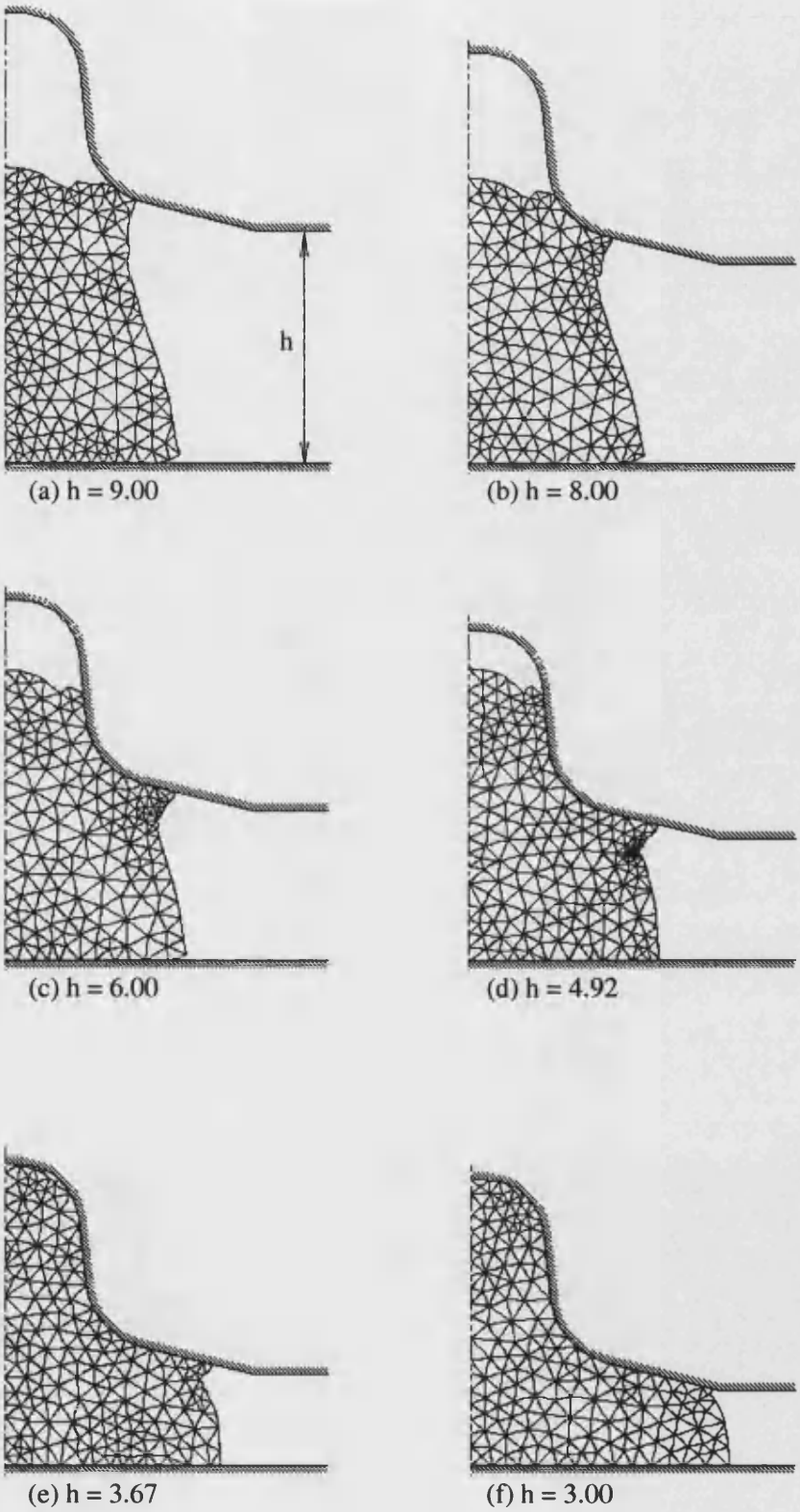


Figure 7.22: Forward simulation using the predicted preform for Case Study 1.

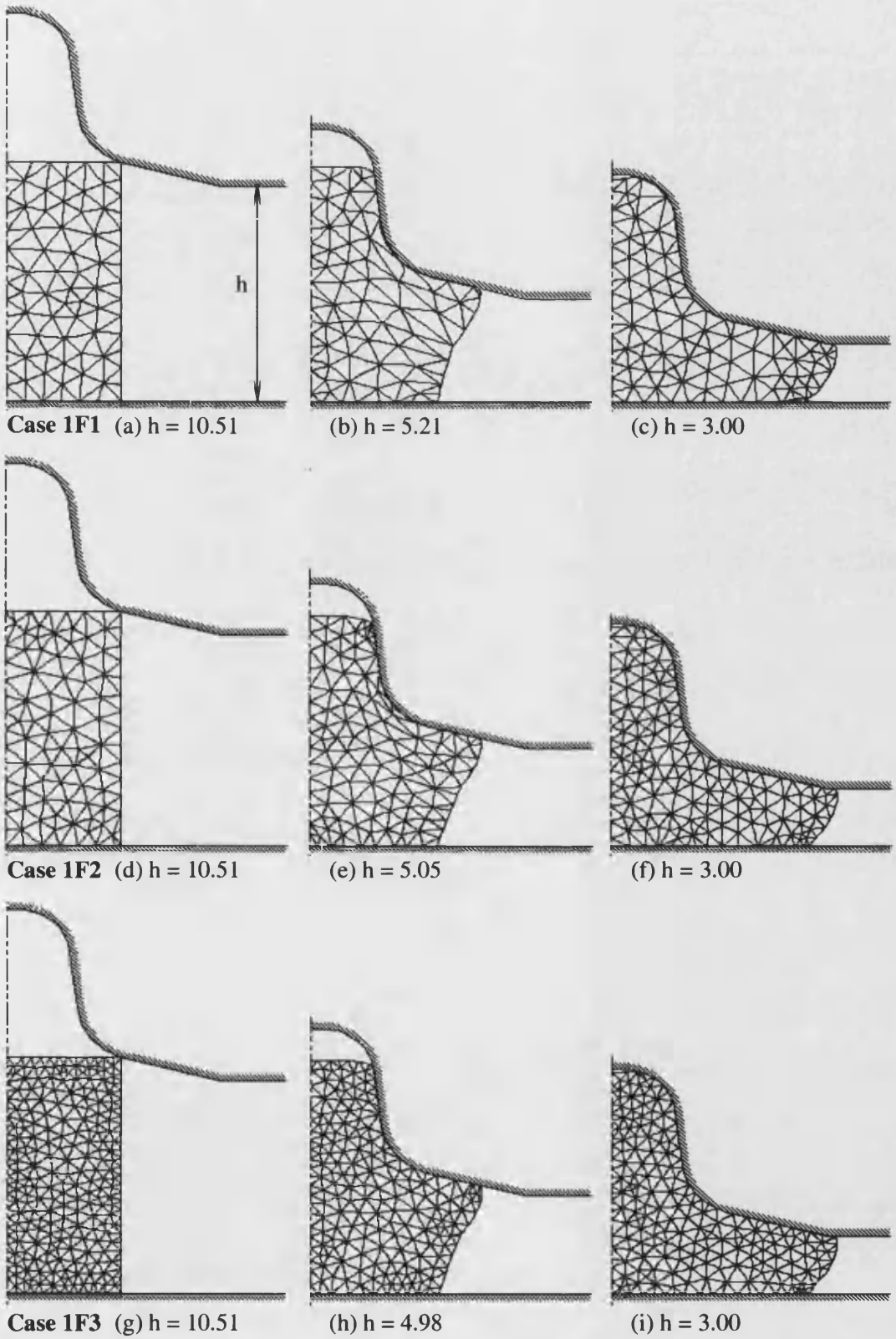


Figure 7.23: Forward simulations resulting from different mesh densities in Case Study 1.

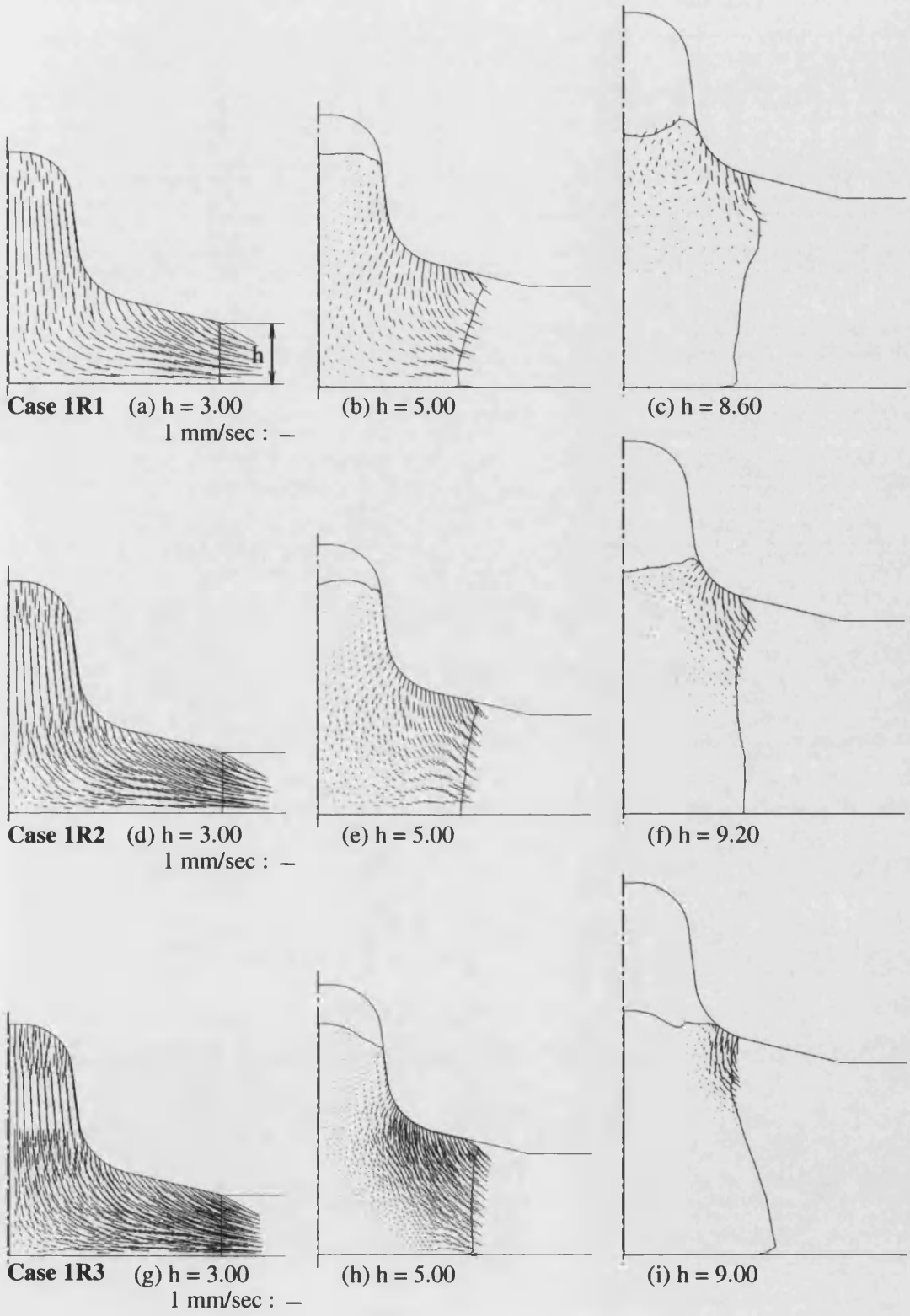


Figure 7.24: Velocity fields resulting from different mesh densities in the reverse simulation of Case Study 1.

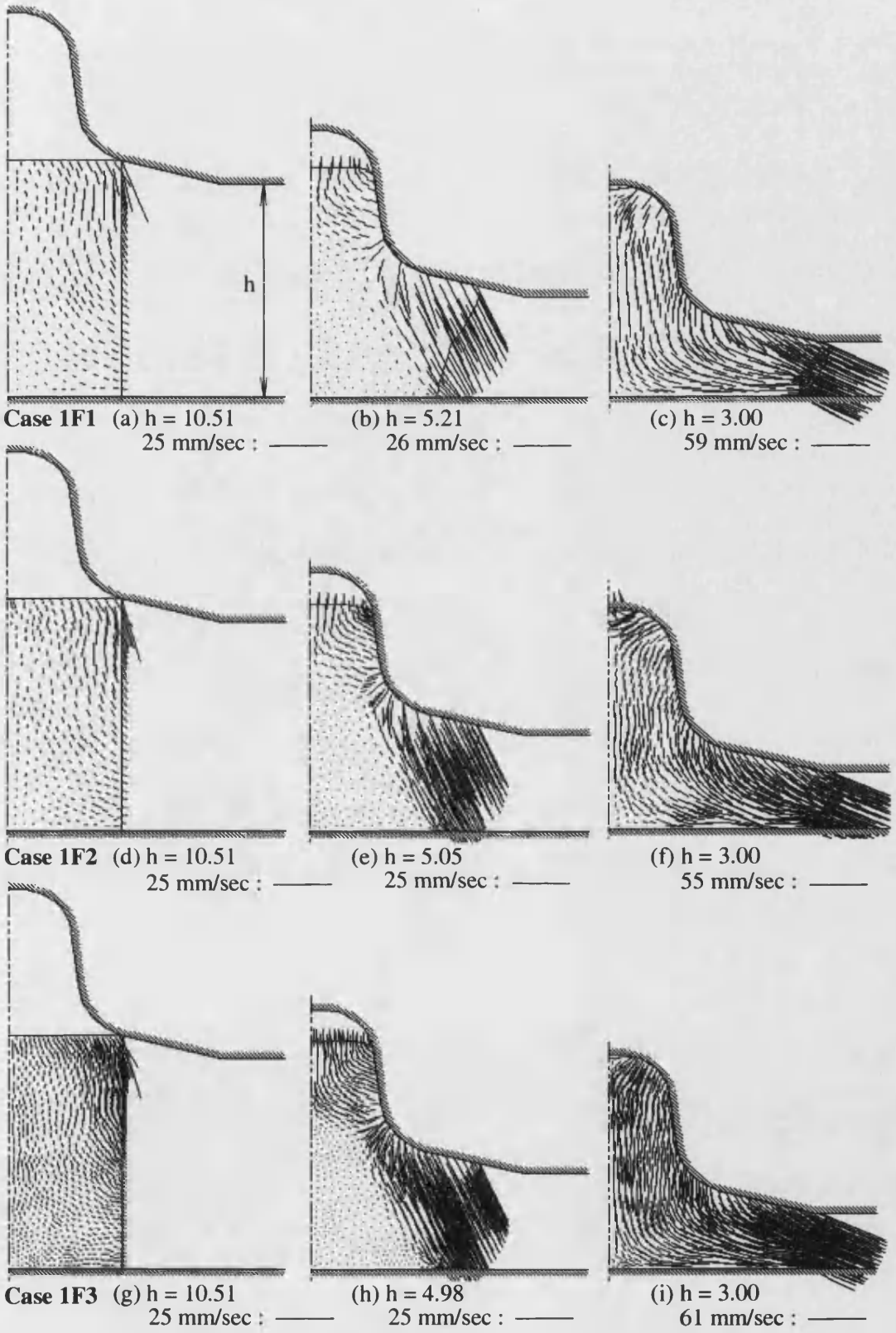


Figure 7.25: Velocity fields resulting from different mesh density in the forward simulation of Case Study 1.

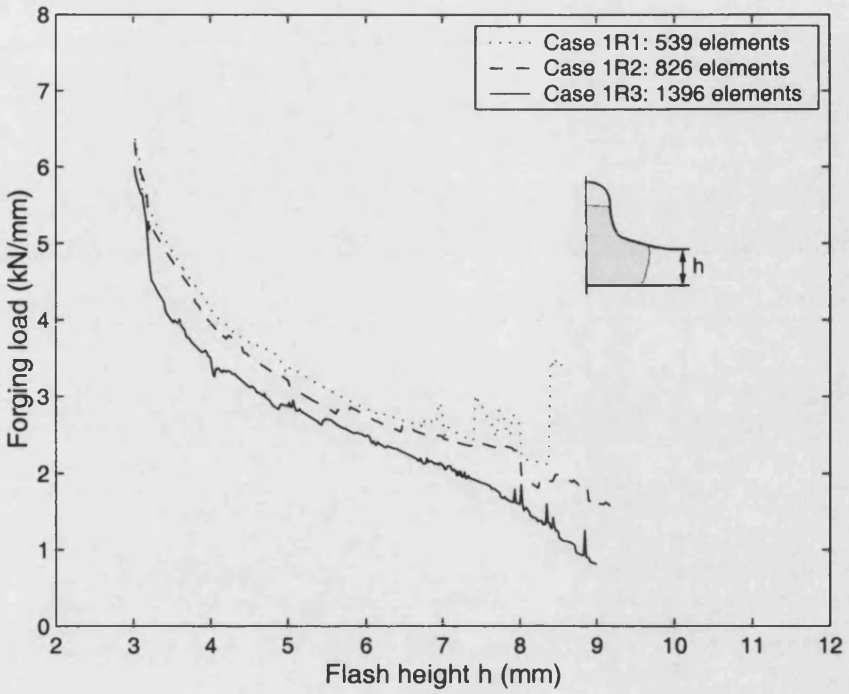


Figure 7.26: Reverse simulation load comparison between different number of elements for Case Study 1.

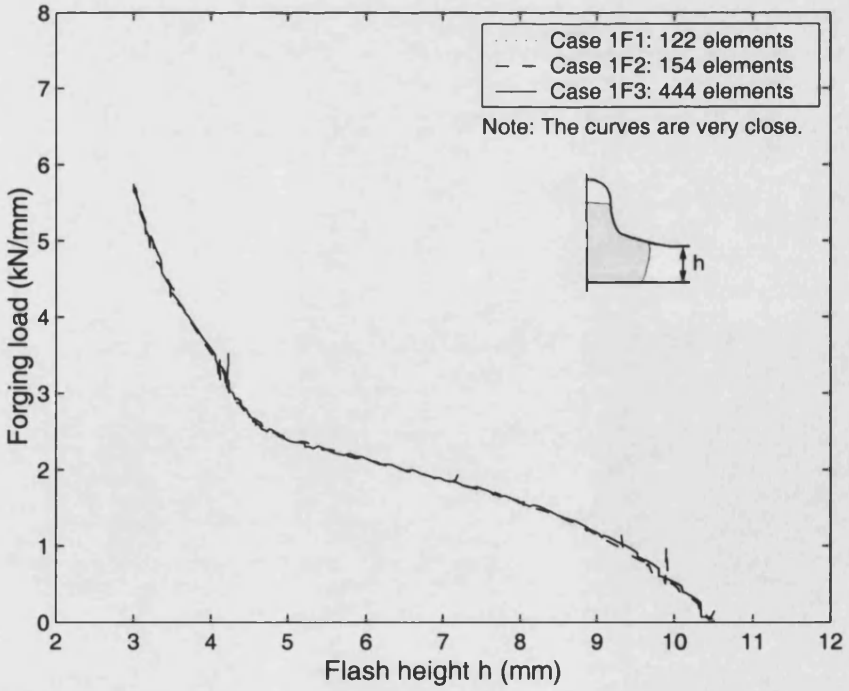


Figure 7.27: Forward simulation load comparison between different number of elements for Case Study 1.

7.4 Case Study 2: Two-cavity Forging

7.4.1 Finished Product Dimensions and Data for Simulations

The second case study considers the two-cavity forging shown in Figure 7.28. Numerical experiments of reverse and forward simulations were carried out to evaluate the proposed preform design approach. The material models, aluminium alloy BS 6063F, were the same as that used in Case Study 1. Table 7.3 provides the details of material models and process conditions for both reverse and forward simulations. Dies were assumed to be rigid in both reverse and forward simulations.

Table 7.3: Material models and process conditions for the reverse and forward simulations of Case Study 2

Simulation type	Workpiece material	Flow stress (MPa)	Tresca friction factor	Die velocity (mm/sec)
Reverse	Rigid-perfectly plastic	176.92	0.3	1
Forward	Strain-hardening	$194.33\bar{\epsilon}^{0.183}$	0.3	-21

7.4.2 Deformation Trend

The material gradually flows into the target shape from the finished shape, Figure 7.30(a), until the bottom die nearly separates from the workpiece to a preform, Figure 7.30(c). After the reverse simulation finished in the first stage, the second

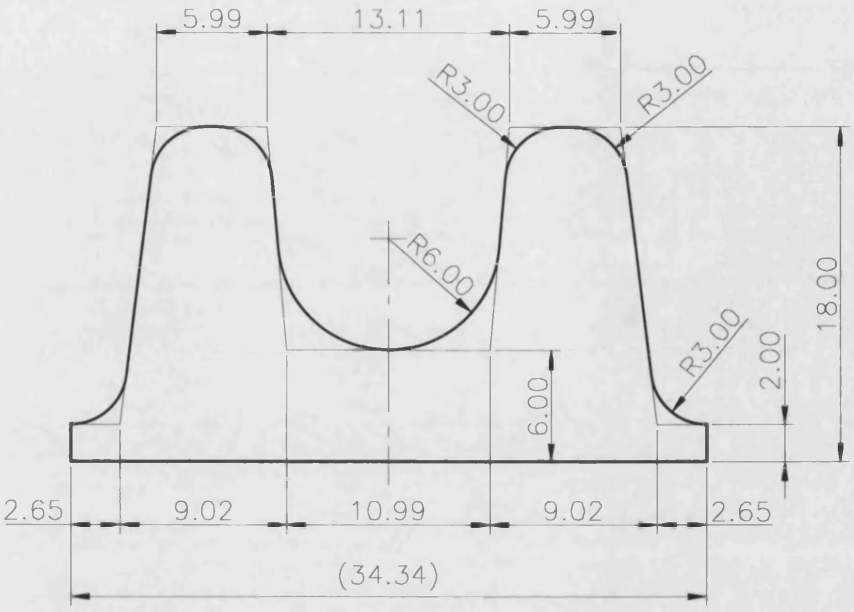


Figure 7.28: Dimensions of Case Study 2.

reverse simulation was needed because the predicted shape was still complex. The adjustment for the jagged boundary was carried out manually using CAD software, AutoCAD [124], to keep the same area as the finished product. The smoothed preform, Figure 7.30(d), was then used to design a new set of dies for the second stage reverse simulation shown in Figure 7.31. Finally, the simulation predicted a simple shape close to a rectangle regarded as the billet when the top die nearly separated from the workpiece. Figure 7.29 shows that the complexity of the workpiece decreased in reverse simulations, and thus also provides the evidence that material flows progressively into the target shape. There is a kink in the first stage at the flash height of 4 mm. The kink could be caused by the re-meshing process which may slightly change the profile of the workpiece.

In both first and second stage reverse simulations, the predicted shapes have jagged boundaries. It seems difficult to use the finite element procedure for producing deformed shapes with smooth boundaries. One of the reasons could be that the 3-noded element formulation for the rigid-perfectly plastic material makes the deformation "rigid", not very flexible. This problem may be solved

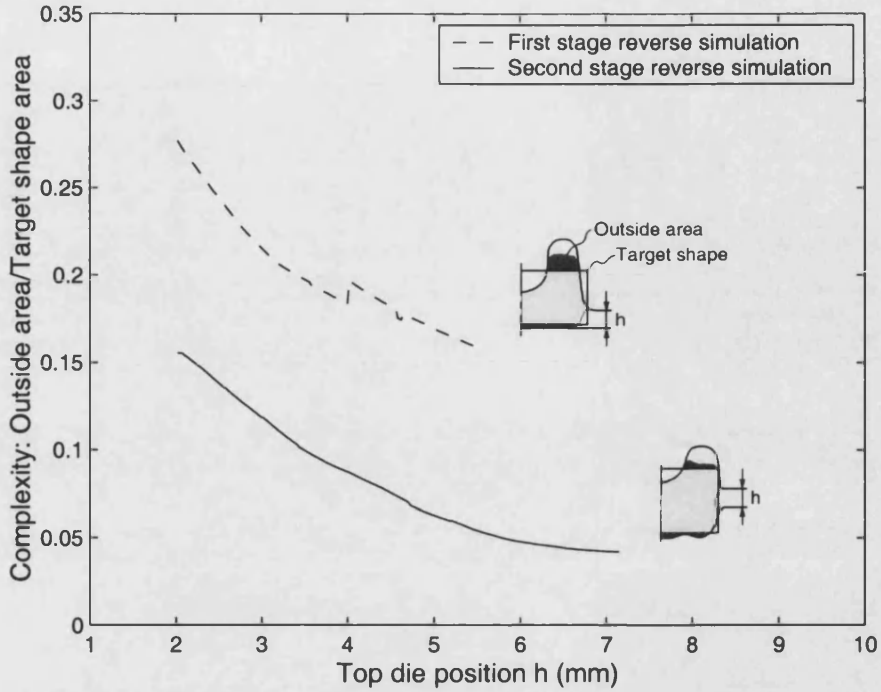


Figure 7.29: Complexity of workpiece during the reverse simulation of Case Study 2.

by using non-linear elements similar to those used in the commercial package FORGE2.

The first stage forward simulation started from the predicted billet, Figure 7.32(a), to a preform, Figure 7.32(d), which was then taken to run the second stage simulation. Figure 7.32(d) shows that there is an unfilled area in the top cavity at the end of the preforming stage. The flash also forms in the gap between dies. Although the preform is different from Figure 7.31(a) predicted by the reverse simulation, a complete die filling product has been predicted as shown in Figure 7.33(d). From Figures 7.33(b) and (c), it can be seen that there is a potential risk to form a fold near the flash in the second stage . This could be caused by the excess flash in the preform, and may be avoided by increasing the height of the billet.

7.4.3 Velocity Field

The velocity field predicted by the first stage reverse simulation shows that the speed of the material flow in the top cavity is relatively high at the end of the process (Figure 7.34(a)). However, in the rest of the reverse increments, the velocity in the same area is relatively small (Figures 7.34(b), (c) and (d)). This result may explain a rapid increase of forging load in Figure 7.38 because a higher velocity field requires more plastic deformation power at the end of the process. A similar phenomenon also exists in the second stage reverse simulation as shown in Figures 7.35 and 7.39.

In the early deformation of the preforming stage, velocity fields shown in Figure 7.35, predicted by the reverse simulation, are slightly different from those shown in Figure 7.36, predicted by the forward simulation. This could be caused by different contact conditions. As the top die approaches the finish position, flow patterns are similar in both simulations (see Figures 7.35(a) and 7.36(d)). In the final stage, velocity fields in the reverse simulation (Figure 7.34) are generally in accord with those in the forward simulation (Figure 7.37), although the reverse simulation predicts a smaller value in the top cavity in comparison with the same deforming region in the forward simulation.

The contact conditions and the velocity at the interface between the workpiece and the die are used to determine the boundary conditions for predicting the velocity field of the deforming workpiece which represents the minimum energy dissipation required for the forging process. The energy dissipation may be expressed in terms of force, i.e. forging load, according to the known velocity of the die. A higher value of the velocity field represents a higher energy dissipation, and therefore a higher forging load is required.

7.4.4 Forging Load

Forging loads predicted by the reverse simulations are generally higher than those predicted by the forward simulations according to the results shown in Figures 7.38 and 7.39. In comparison with the maximum forging load, the reverse simulations clearly predict higher values in both stages, especially in the preforming stage, Figure 7.39. An unfilled area in the top cavity shown in Figure 7.32(d) could be the reason for the significant difference of the maximum load. Furthermore, in Figure 7.38, a kink of forging load at the flash height of around 5.5 mm may be caused by the situation that the flash of the preform contacts the top die in a small area at the early stage of the deformation as shown in Figure 7.33(a).

Forging loads resulting from plastic deformation and friction in reverse simulations are presented in Figures 7.40 and 7.41. As the top die moves upwards, the contact area between dies and the workpiece decreases, during the reverse simulation. Friction effects and forging load resulting from friction thus gradually reduce as the flash height increases as shown in Figure 7.40. However, in the second stage reverse simulation, friction does not make a significant effect on forging load as shown in Figure 7.41 although the contact area is also large in early deformation. This result may be due to the magnitude of the tangential velocity along the die surface being small in most of the deformation in the preforming process, especially at the bottom-right corner as shown in Figures 7.35 and 7.36.

7.4.5 Possibility of Using One-stage Forging

To investigate the possibility of using one-stage forging process to produce the desired product in Case Study 2, three different billets were simulated by the

commercial package FORGE2. The sizes of the billets and the results of the simulations are shown in Table 7.4 and Figure 7.42, respectively. It can be seen that Case 2A in Figure 7.42 is not completely filled. Case 2B and Case 2C however fulfil the requirement of the complete die filling as shown in Figures 7.42(f) and (i). Figure 7.43 shows that the maximum forging load is close to 20 (kN/mm) in both Case 2B and Case 2C. The unit of forging load is force per length because of the condition of plane strain. The total dissipation energy of Case 2B is 43.82 (J/mm) and lower than 52.77 (J/mm) of Case 2C. Case 2B with complete filling and a lower total energy dissipation was therefore compared with the two-stage forging process predicted by the reverse simulation. Figure 7.44 shows that the maximum forging load is close to 20 (kN/mm) in both cases. The total dissipation energy of the two-stage process 47.89 (J/mm) is however slightly higher than that of the one-stage forging 43.82 (J/mm). This could be due to the fact that the flash formed in Figure 7.32(d) increases the required forging load.

From the results of this case study, the reverse simulation could predict preforms which might not be required. This issue which involves the reversibility of metal flow, the numerical issues on element type and the releasing node strategy, may need further study and investigation by means of experiments.

Table 7.4: Variation of billet sizes for one-stage forging in Case Study 2

Test	Billet size (width × height)	Results Figure 7.42	Complete die filling
Case 2A	12.00 × 15.27	(a)-(c)	NO
Case 2B	10.00 × 18.32	(d)-(f)	YES
Case 2C	8.00 × 22.90	(g)-(i)	YES

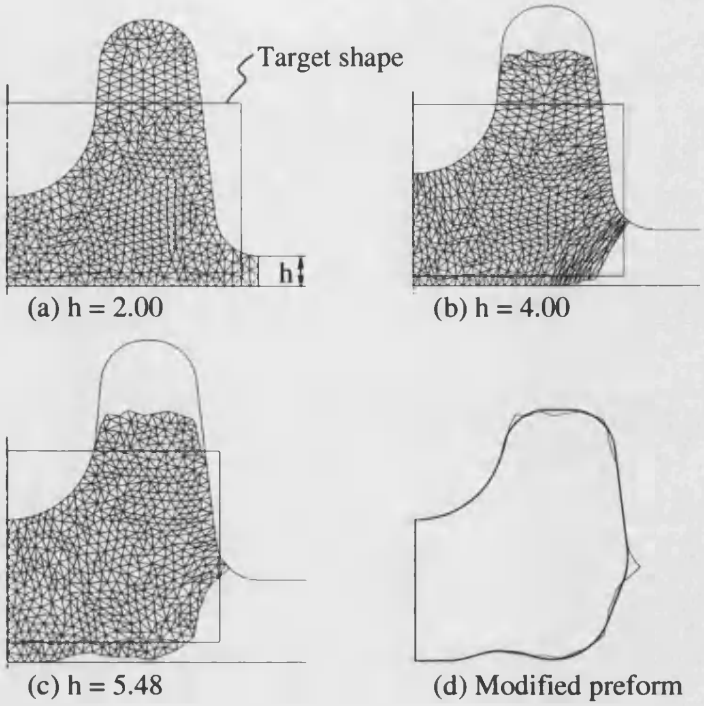


Figure 7.30: First stage reverse simulation of Case Study 2.

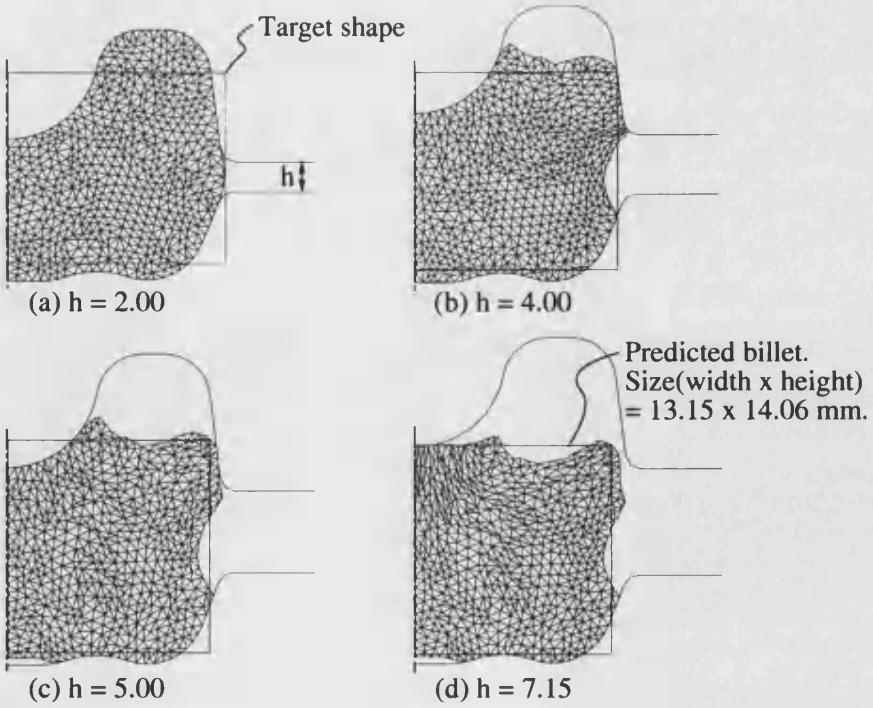


Figure 7.31: Second stage reverse simulation of Case Study 2.

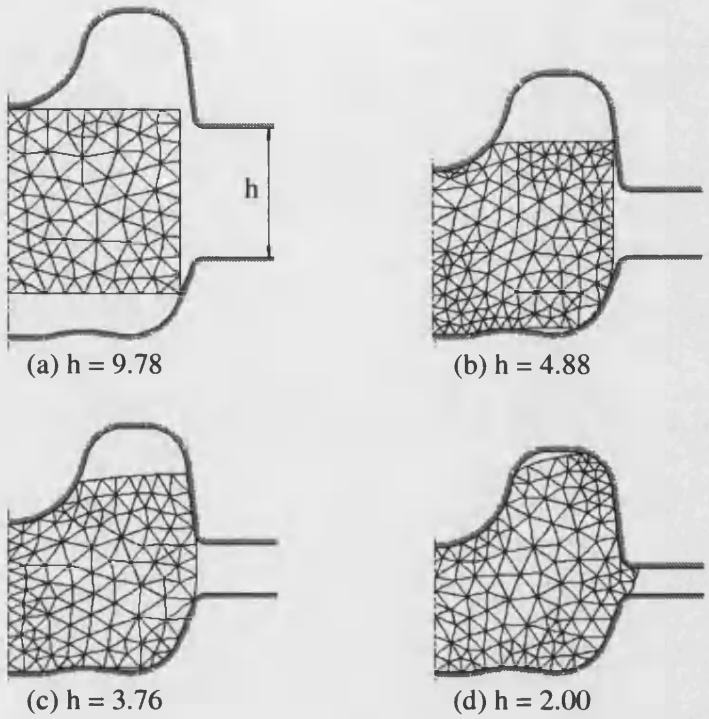


Figure 7.32: First stage forward simulation of Case Study 2.

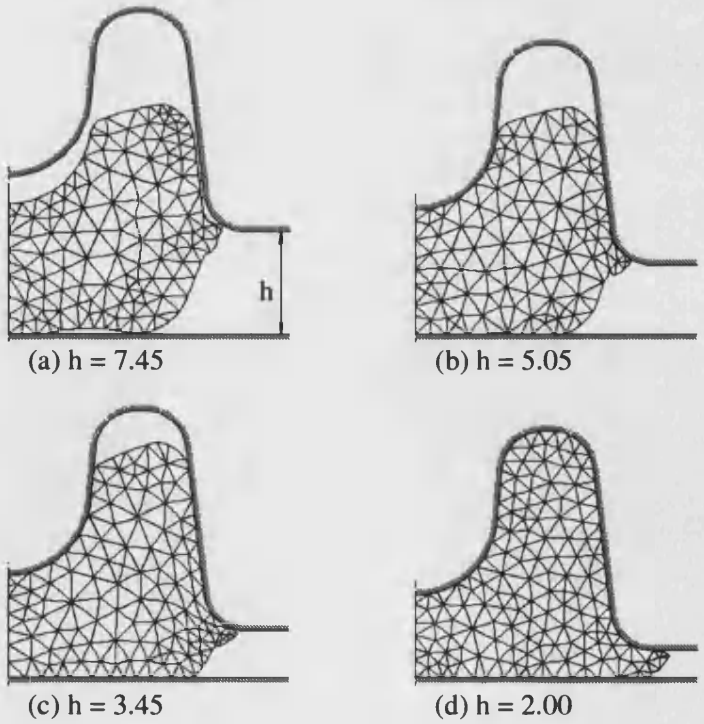


Figure 7.33: Second stage forward simulation of Case Study 2.

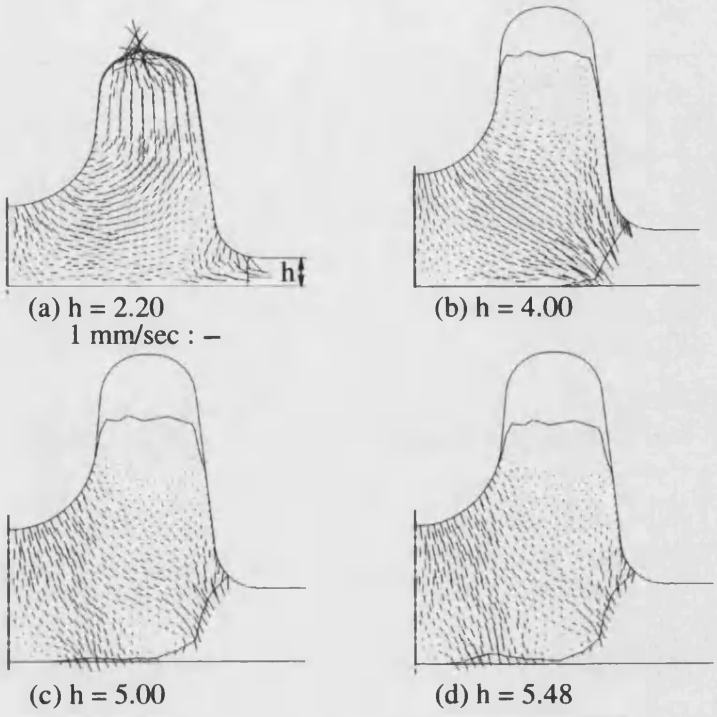


Figure 7.34: Velocity fields in the first stage reverse simulation of Case Study 2.

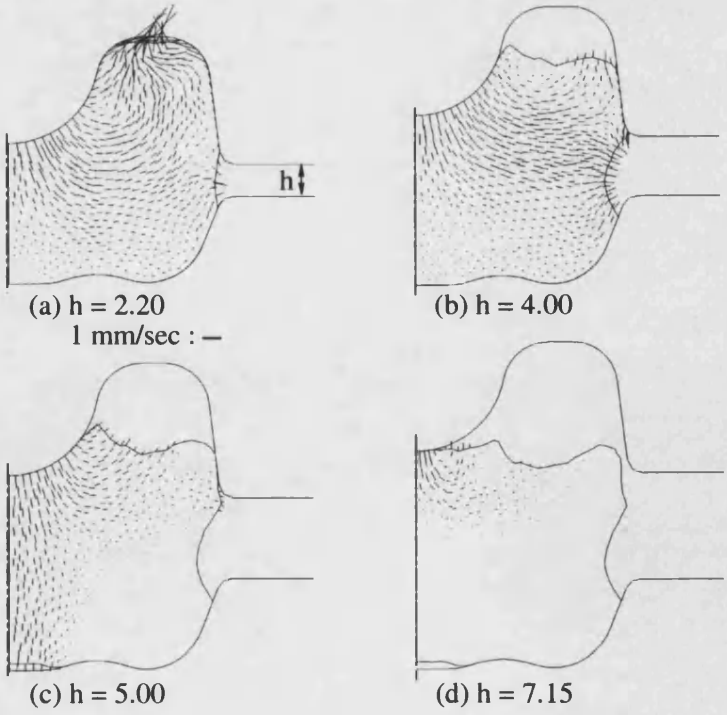


Figure 7.35: Velocity fields in the second stage reverse simulation of Case Study 2.

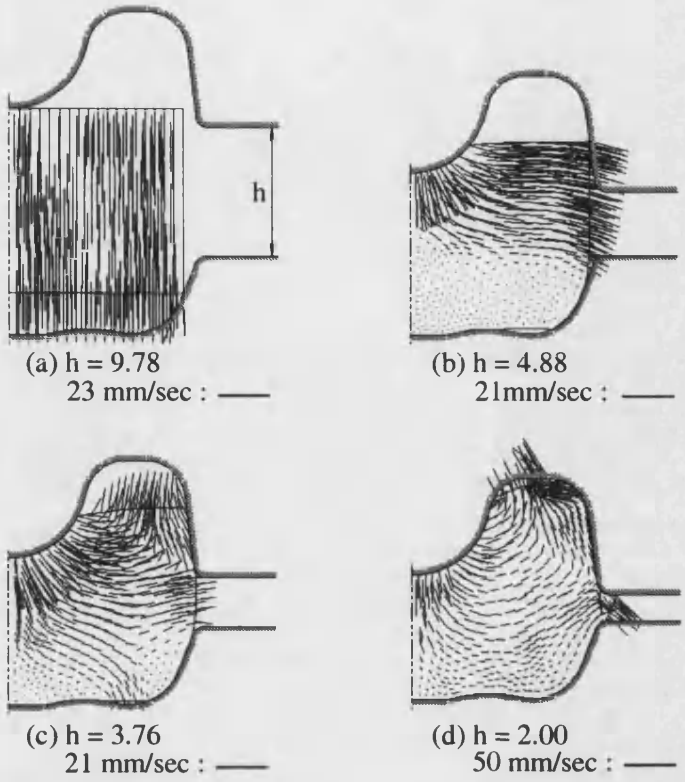


Figure 7.36: Velocity fields in the first stage forward simulation of Case Study 2.

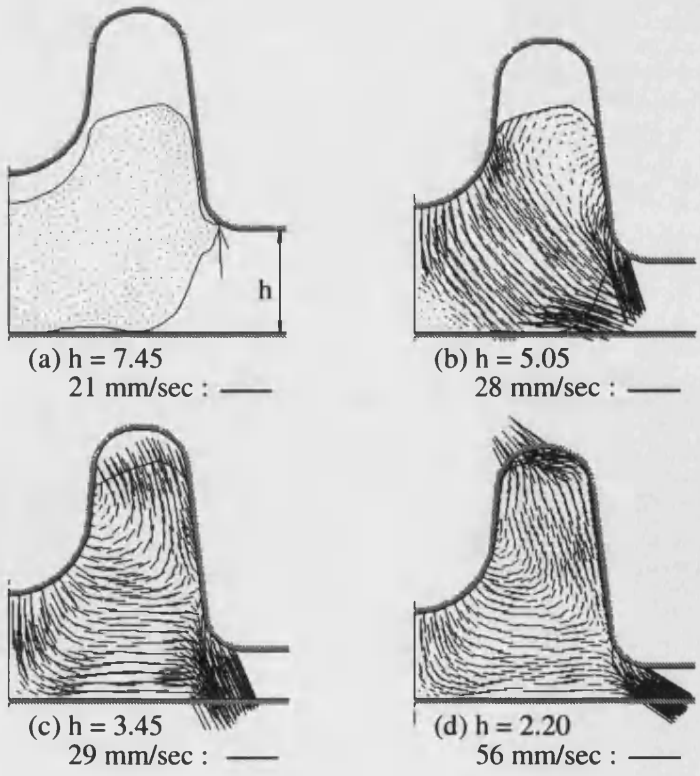


Figure 7.37: Velocity fields in the second stage forward simulation of Case Study 2.

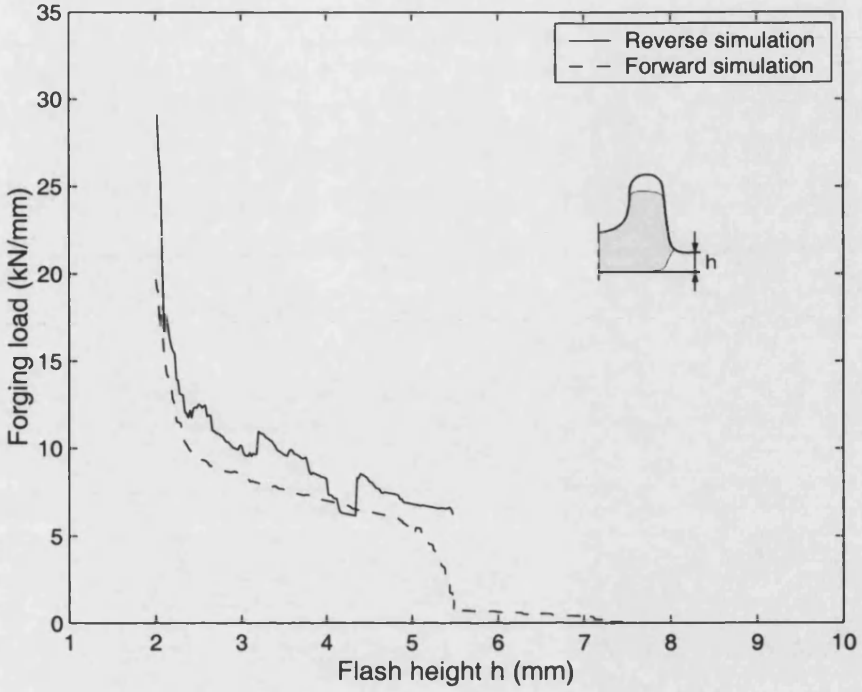


Figure 7.38: Forward and reverse load predictions for the first stage of Case Study 2.

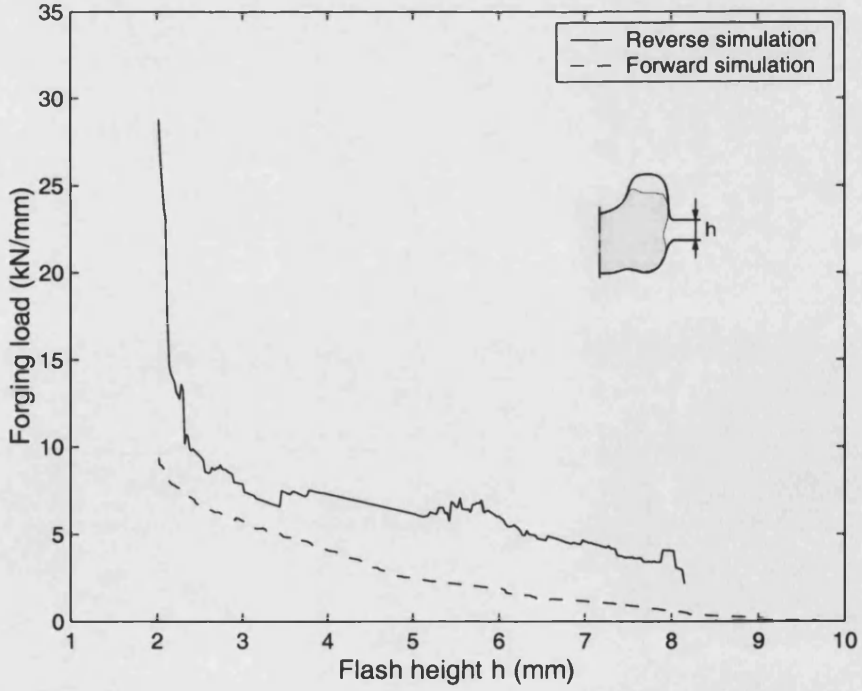


Figure 7.39: Forward and reverse load predictions for the second stage of Case Study 2.

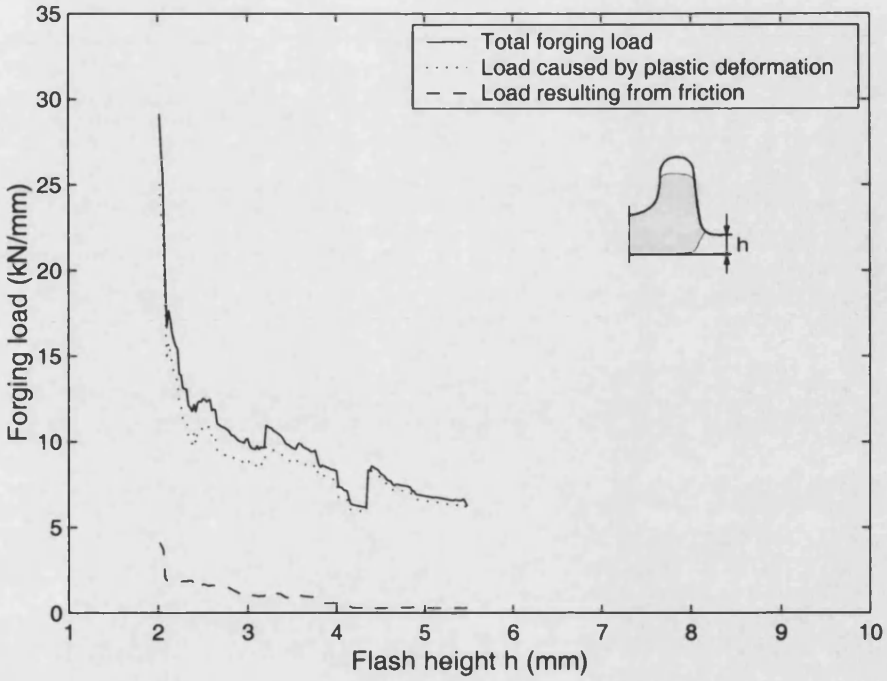


Figure 7.40: Load predictions for the first stage reverse simulation of Case Study 2.

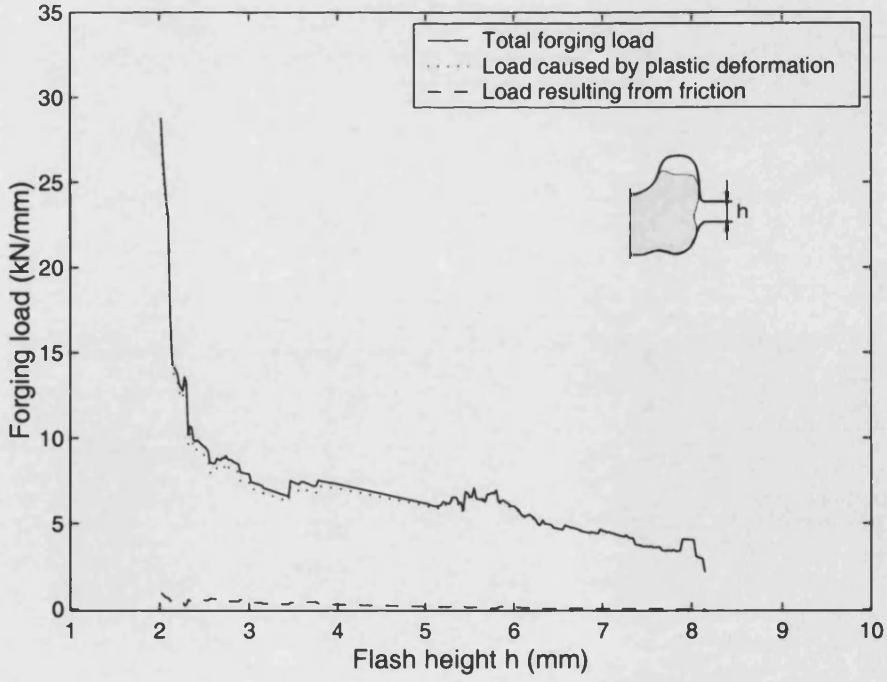


Figure 7.41: Load predictions for the second stage reverse simulation of Case Study 2.

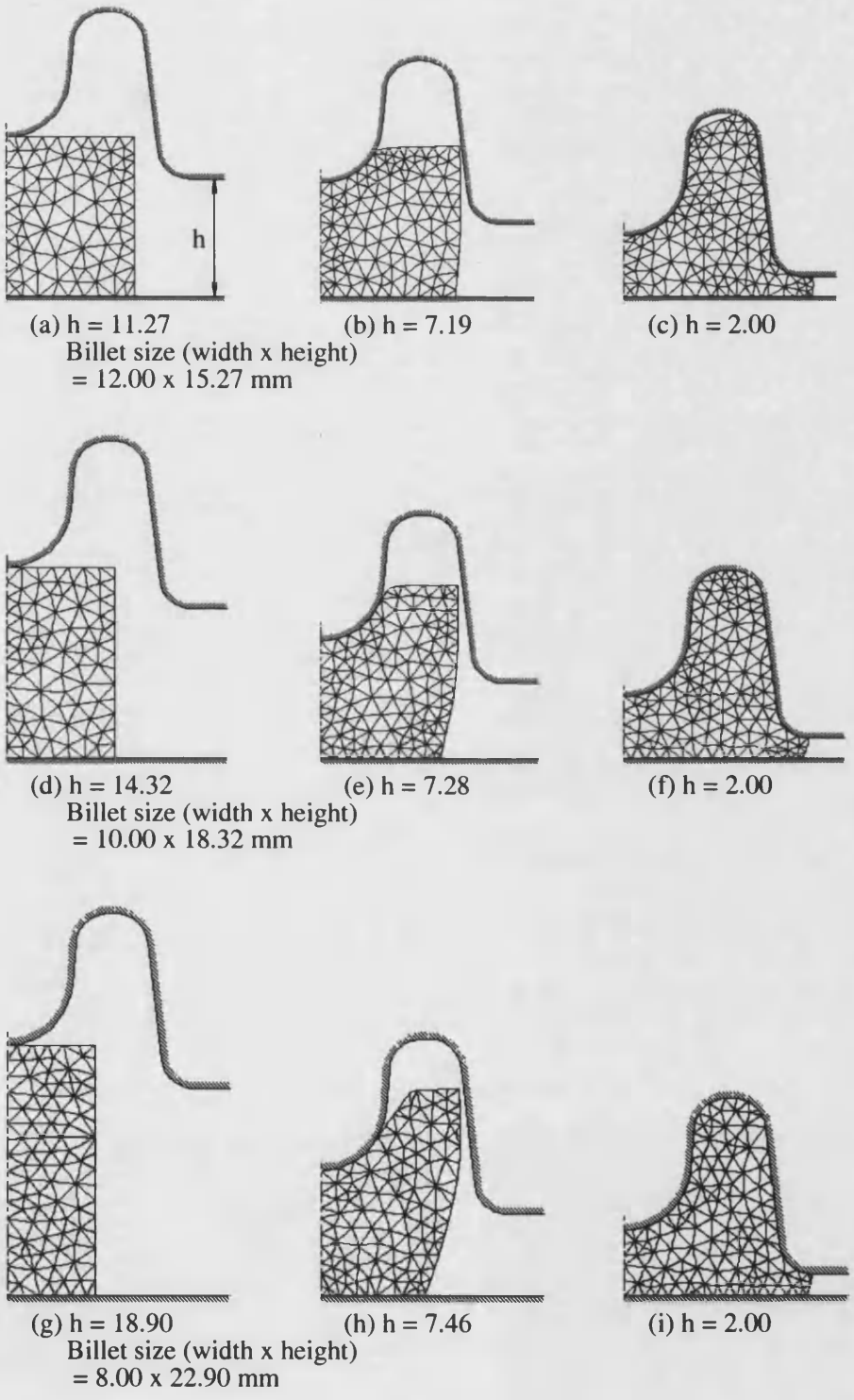


Figure 7.42: Forward simulations by using one-stage forging with different billet sizes for Case Study 2.

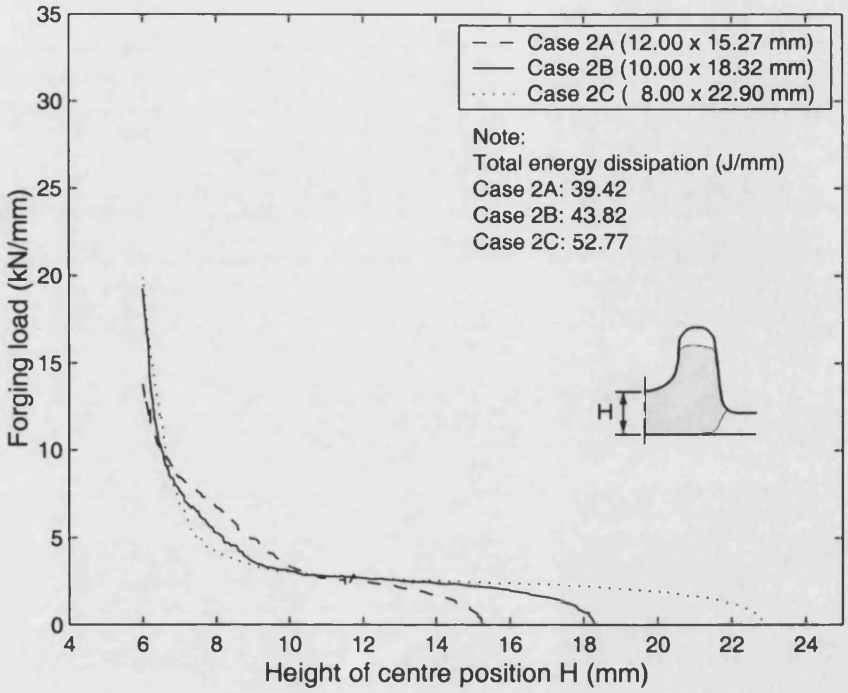


Figure 7.43: Forward load predictions for the one-stage forging simulation using different billet sizes for Case Study 2.

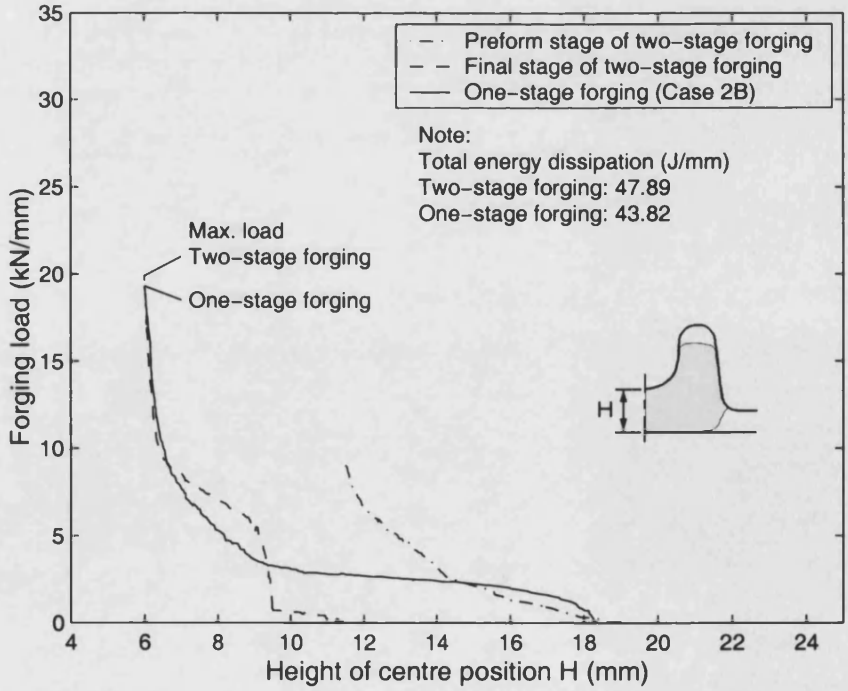


Figure 7.44: Load predictions for the one-stage and two-stage forward simulations of Case Study 2.

7.5 Case Study 3: Three-cavity Forging

7.5.1 Finished Product Dimensions and Data for Simulations

The third case study is a three-cavity forging shown in Figure 7.45. The material models and the process conditions for this case study shown in Table 7.5 were the same as those used in Case Study 2. The material of the dies was assumed to be rigid in both reverse and forward simulations.

Table 7.5: Material models and process conditions for the reverse and forward simulations of Case Study 3

Simulation type	Workpiece material	Flow stress (MPa)	Tresca friction factor	Die velocity (mm/sec)
Reverse	Rigid-perfectly plastic	176.92	0.3	1
Forward	Strain-hardening	$194.33\bar{\epsilon}^{0.183}$	0.3	-21

7.5.2 Deformation Trend

The reverse simulation started from the finished shape shown in Figure 7.47(a) until the bottom die nearly separated from the workpiece. The predicted shape Figure 7.47(c) was still complex and a further reverse simulation was therefore needed. A smoothed preform shown in Figure 7.47(d) was determined manually using AutoCAD, and was used to design a new set of die profiles for the second stage reverse simulation shown in Figure 7.48. The predicted preform was still

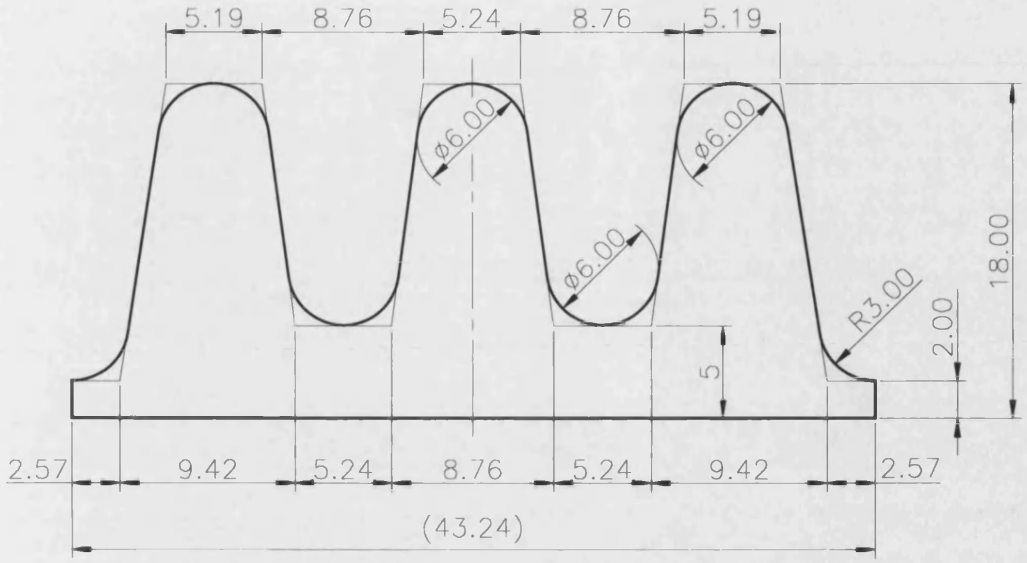


Figure 7.45: Dimensions of Case Study 3.

complex after two reverse simulations and a similar procedure including smoothing boundaries and determining new dies was carried out for the third stage of the reverse simulation. The predicted deformation is shown in Figure 7.49. Finally the reverse simulation predicted a simple shape close to a rectangle regarded as the billet when the top die nearly separated from the workpiece.

Figures 7.47, 7.48 and 7.49 show that the material flows gradually towards the target shape as the dies move backwards in the reverse simulations. This movement can also be verified in terms of the decreasing complexity of the shape, shown in Figure 7.46. However, the predicted deformation has concave profiles on free surface instead of convex profiles which are expected in practical forging processes. The forging process might not be traced in reverse although the backwards tracing scheme has been used in the modelling. The issue of the reversibility should be investigated in future studies, especially for more complex cases. As in Case Study 2, smooth boundaries of the workpiece are difficult to predict. A higher order interpolation of the velocity field, such as using 6-noded elements, may predict smoother boundaries of the workpiece.

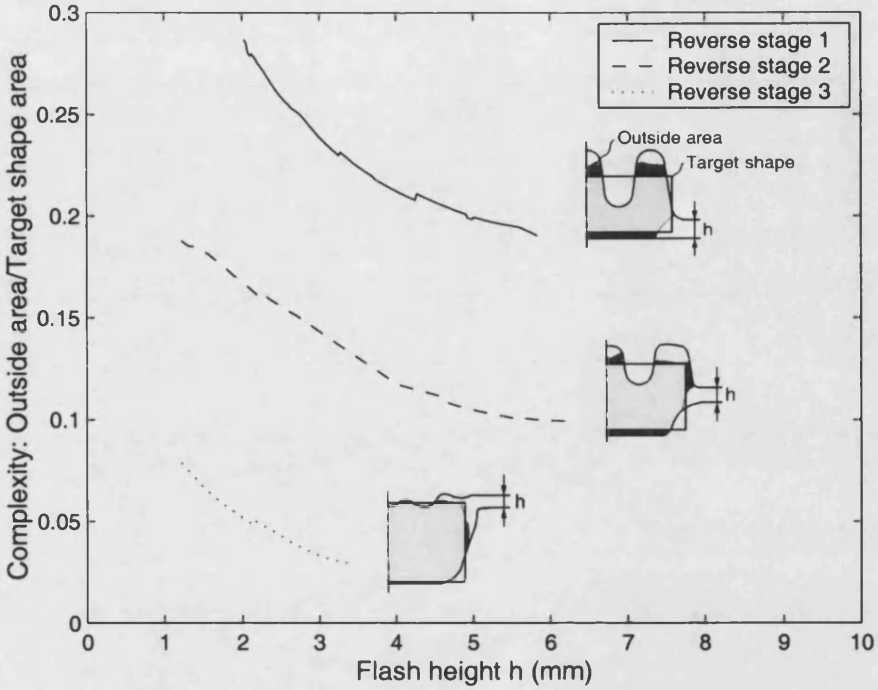


Figure 7.46: Complexity of workpiece during the reverse simulation of Case Study 3.

The first stage forward simulation started from the predicted billet shown in Figure 7.50(a) and finished at a preform shown in Figure 7.50(d). This preform was then taken to run the second stage simulation and the second preform, shown in Figure 7.51(d), was obtained. An unfilled area in the top-right cavity occurs and a potential fold may develop in this area at this stage. The following stage, the third stage, forward simulation finally finished with the result of complete die filling (Figure 7.52(d)). The predicted results, Figures 7.52(b), (c) and (d), show that potential folds may develop in the right cavity and the flash area. This could be caused by the concave profiles predicted in preform.

7.5.3 Velocity Field

The velocity of material flows gradually decreases from the beginning to the end of the forging process, i.e. from Figure 7.53(a) to Figure 7.53(d), in the first

stage reverse simulation. This phenomenon has also been found in the second and third stage reverse simulations, and is consistent with the results of the forward simulations shown in Figures 7.56, 7.57 and 7.58. The predicted velocity field also corresponds to forging process. Since the die velocity is constant, the velocity of material flow increases as the amount of free surface or flash area decreases. Moreover, the results of the reverse simulations reveal some deformed zones with low velocity and are generally in accord with those of the forward simulations.

7.5.4 Forging Load

Forging loads shown in Figures 7.59, 7.60 and 7.61 decrease as the respective flash heights increase in both reverse and forward simulations. This phenomenon corresponds to the required power of the deformation in the velocity fields from a high value of flow speed to a low value. In comparison with the maximum forging load, the reverse simulations predict higher values in all stages, especially in the second stage shown in Figure 7.60. An unfilled area in the top-right cavity shown in Figure 7.51(d) could be the reason for the significant difference of the maximum load. Moreover, the maximum forging load predicted by the second stage reverse simulation is higher than that predicted by the first stage. As the dies gradually separate from the workpiece, the contact area between the dies and the workpiece reduces, and thus friction effects decrease. This can be seen from Figures 7.62 and 7.63 in which forging load caused by friction gradually decreases. In the third stage of the reverse simulation, the magnitude of tangential velocity along the die surfaces is small and thus the forging load caused by friction is not significant. This is shown in Figure 7.64.

7.5.5 Possibility of Using One-stage Forging

Three different billets listed in Table 7.6 were used to investigate whether there are possibilities of forming the desired shape of Case Study 3 by using a one-stage forging process. The predicted results in Figure 7.65 show that complete die filling can be achieved if the billet size is carefully selected, e.g. Case 3B and 3C. The required forging load and the total dissipation energy are presented in Figure 7.66. Case 3B with a complete die filling and a lower total dissipation energy was therefore compared with the forward simulation of the three-stage forging process which was derived from the predicted preforms of the reverse simulation. Figure 7.67 shows that the maximum forging load is close to 27 (kN/mm) but the one-stage case has a slightly higher value. However, the three-stage forging process consumes more energy than the one-stage forging. Moreover, a simulation starting from the second stage preform was also carried out. The results shown in Figure 7.68 reveal that it is possible to achieve complete die filling.

The above results could imply that (1) the reverse simulation may not predict the optimal preforms under the consideration of minimum total dissipation energy; and (2) the required number of the preforms could be less than that predicted by the reverse simulation in which the achievement of complete die filling is the objective. The releasing node scheme, which assumes the detachment of the furthest node on a die surface to the target shape, may not result in the optimal boundary conditions to achieve the minimum dissipation energy or the shortest

Table 7.6: Variation of billet sizes for one-stage forging in Case Study 3

Test	Billet size (width × height)	Results Figure 7.65	Complete die filling
Case 3A	14.00 × 16.94	(a)-(c)	NO
Case 3B	12.00 × 19.77	(d)-(f)	YES
Case 3C	10.00 × 23.72	(g)-(i)	YES

path of the deformation from the billet to the desired shape. Moreover, the rectangular target shape, i.e. the simplest shape, could be too different from the deformed workpiece which is a more complex shape. Image metamorphosis, or image morphing [125, 126, 127, 128] , which is a technique for creating a smooth transition between two images, could be used to define a non-rectangular target shape as shown in Figure 7.69. The shape could be an intermediate shape (or image) with a level of complexity closer to the deformed workpiece, such as the 80% difference shown in Figure 7.69(c). This is one of the tasks in future studies to improve the accuracy of the reverse simulation.

7.6 Summary

Studies including both practical experiments and numerical simulations have been carried out to evaluate the proposed preform design approach which uses the reverse simulation with the proposed contact releasing method. During the reverse simulations, material flows gradually move into the target shape and the shape of the workpiece varying accordingly from "complex" to "simple", and the preforms are finally predicted.

Case Study 1 shows that the modelling of forging process using the upper bound based reverse simulation with finite element procedure is able to predict the billet for this one-stage forging. The trend of the deformation is close to that of the experiments although some areas of the predicted deformation are different. The forward simulations carried out by the commercial package, FORGE2, also predict sensible results in comparison with the experiments. The forging load predicted by both types of the simulations, especially that predicted by the reverse simulation, is generally higher than that acquired from the experiments.

Case Studies 2 and 3, the multi-cavity forgings, demonstrate that the proposed design approach is able to achieve the requirement of complete die filling. The case studies also reveal the possibility of reducing the number of preforms predicted by the reverse simulation. The deformation trend of the reverse simulation is similar to that of forward simulation although smoother boundaries of the workpiece are predicted by the latter. Moreover, the forging load estimated by the reverse simulation is higher that calculated by forward simulation.

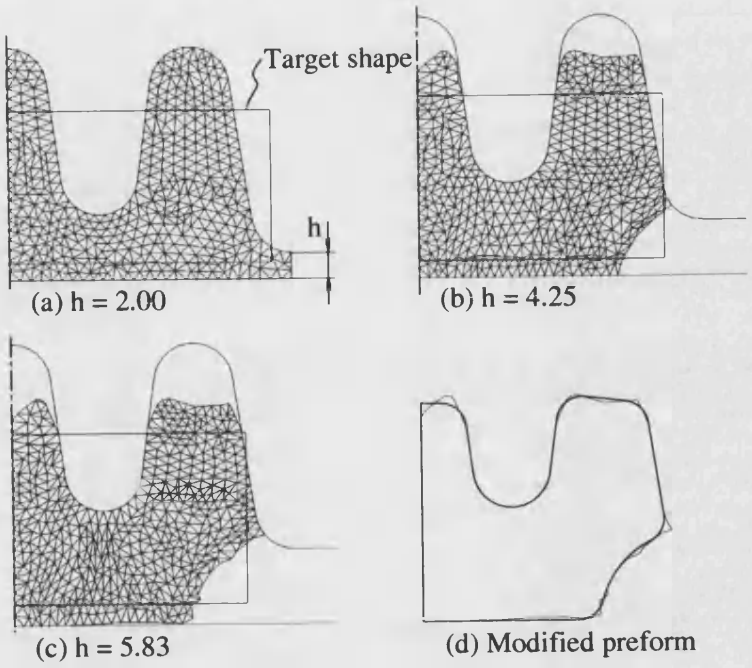


Figure 7.47: First stage reverse simulation of Case Study 3.

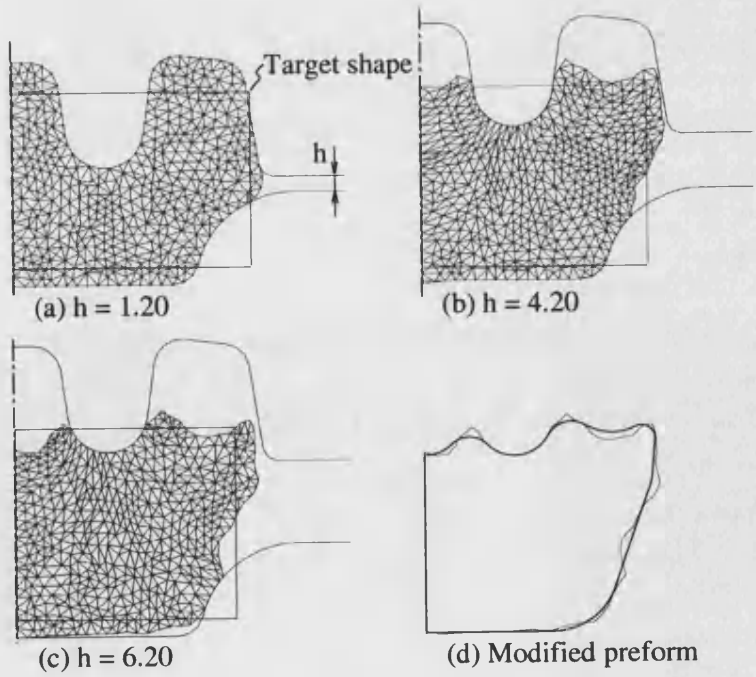


Figure 7.48: Second stage reverse simulation of Case Study 3.

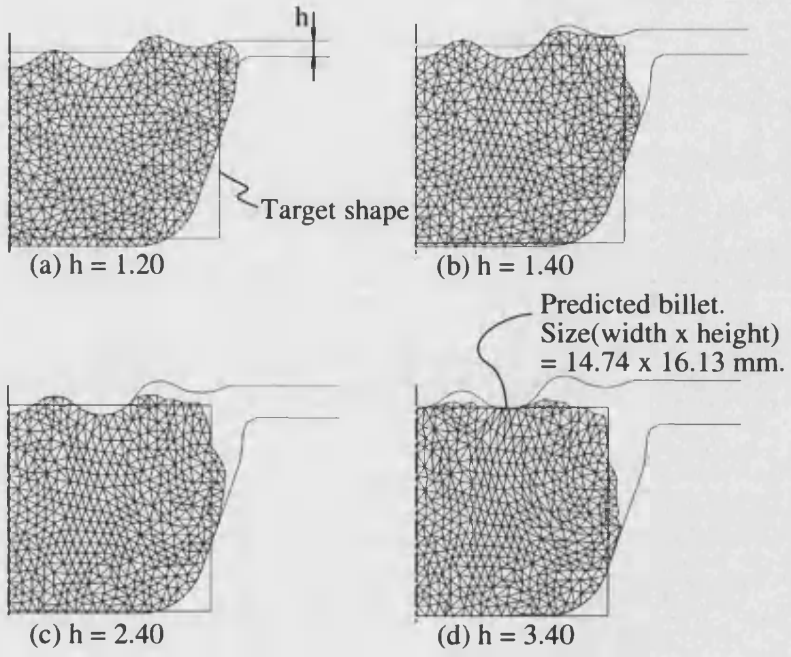


Figure 7.49: Third stage reverse simulation of Case Study 3.

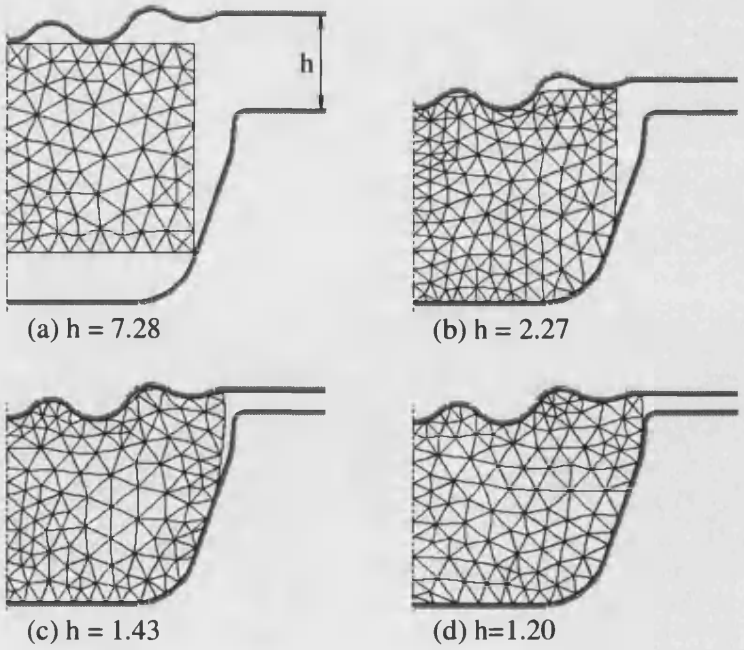


Figure 7.50: First stage forward simulation of Case Study 3.

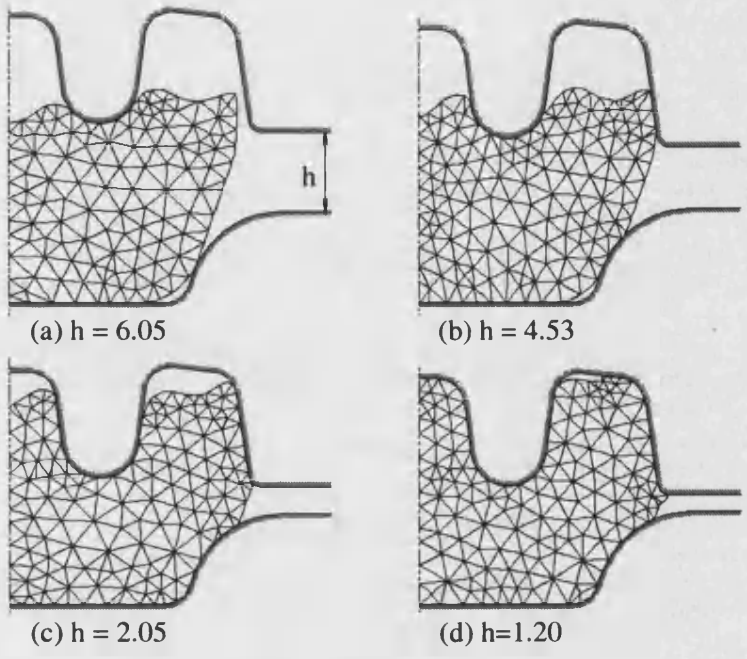


Figure 7.51: Second stage forward simulation of Case Study 3.

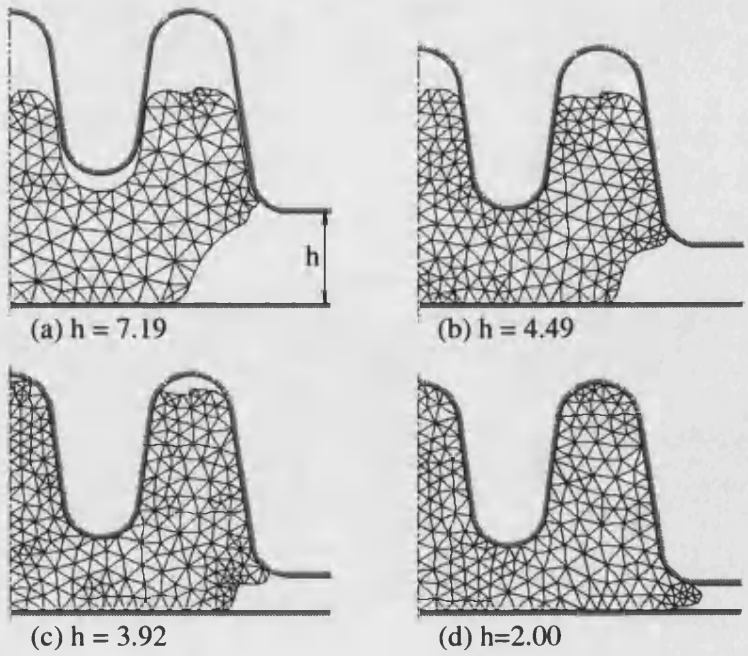


Figure 7.52: Third stage forward simulation of Case Study 3.

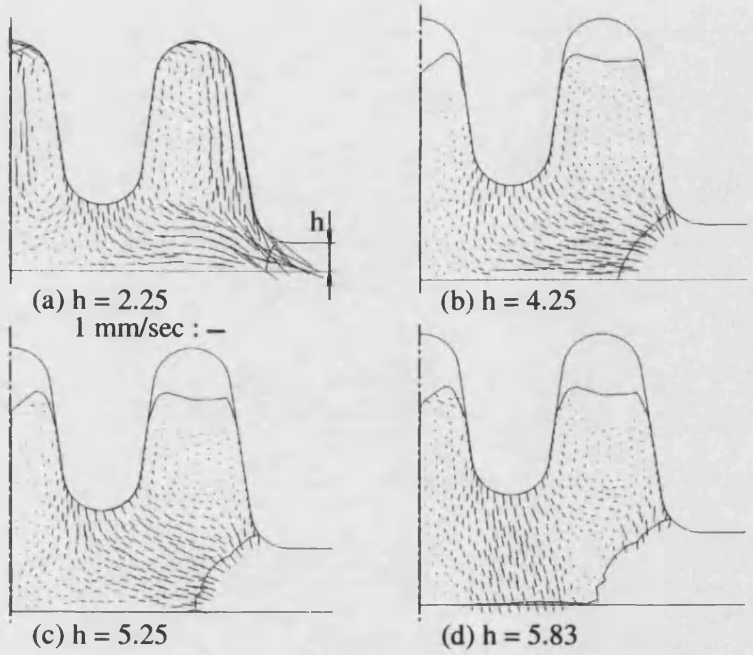


Figure 7.53: Velocity fields in the first stage reverse simulation of Case Study 3.

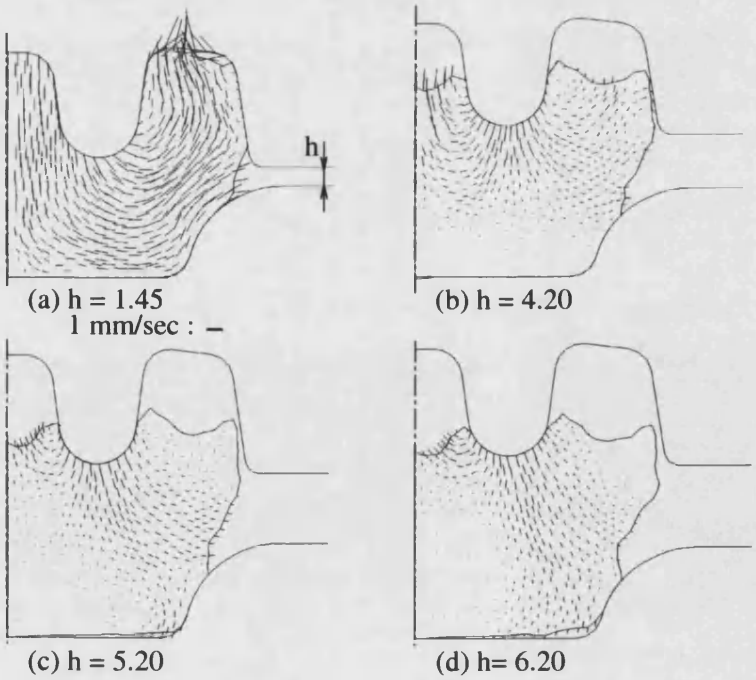


Figure 7.54: Velocity fields in the second stage reverse simulation of Case Study 3.

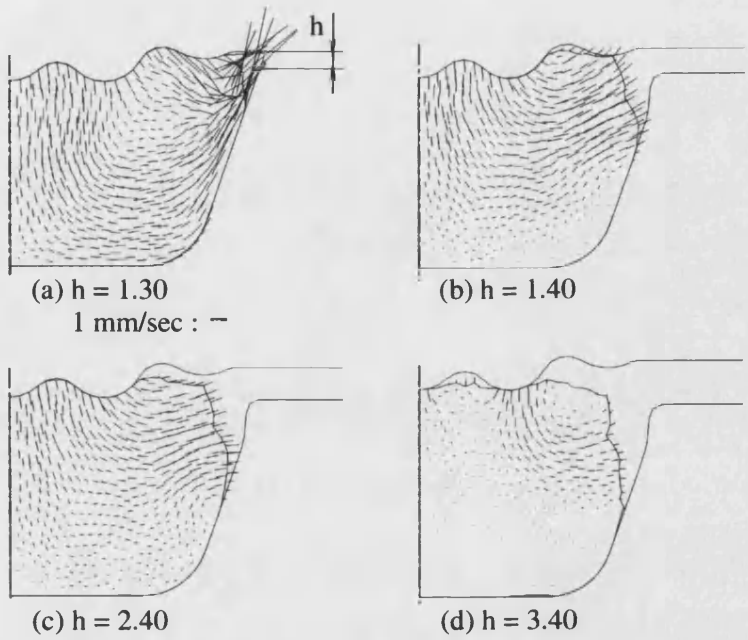


Figure 7.55: Velocity fields in the third stage reverse simulation of Case Study 3.

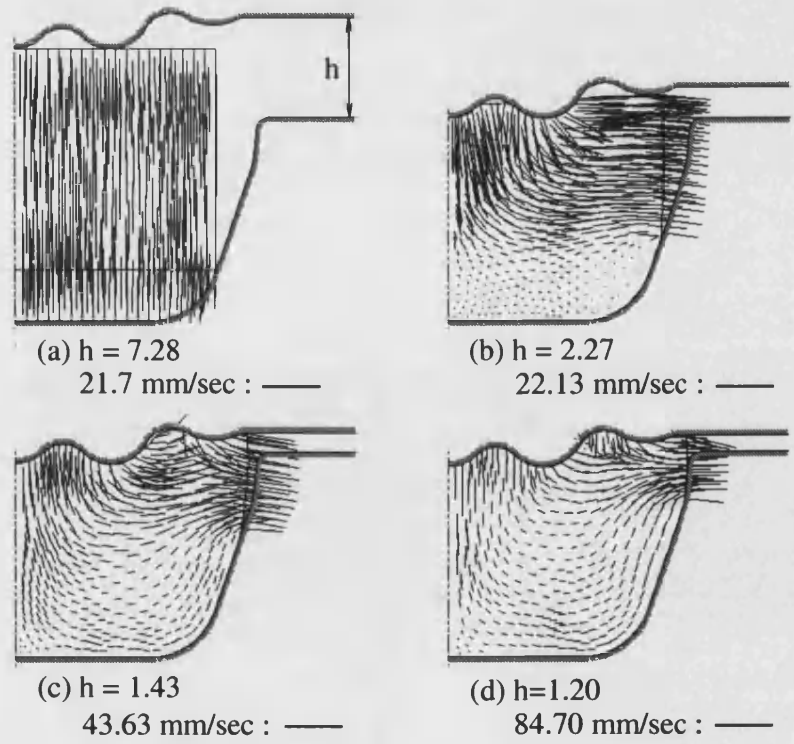


Figure 7.56: Velocity fields in the first stage forward simulation of Case Study 3.

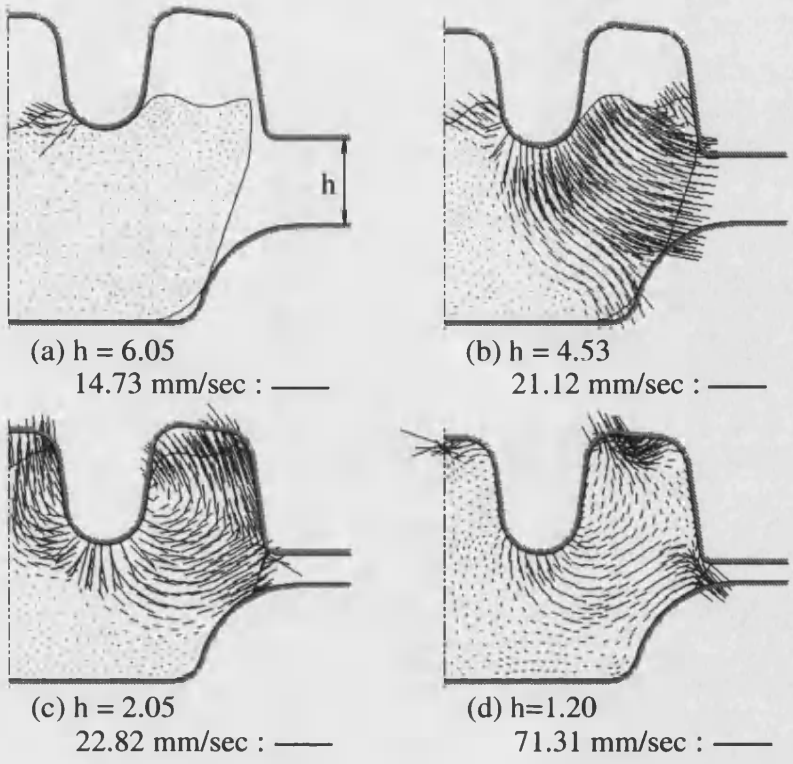


Figure 7.57: Velocity fields in the second stage forward simulation of Case Study 3.

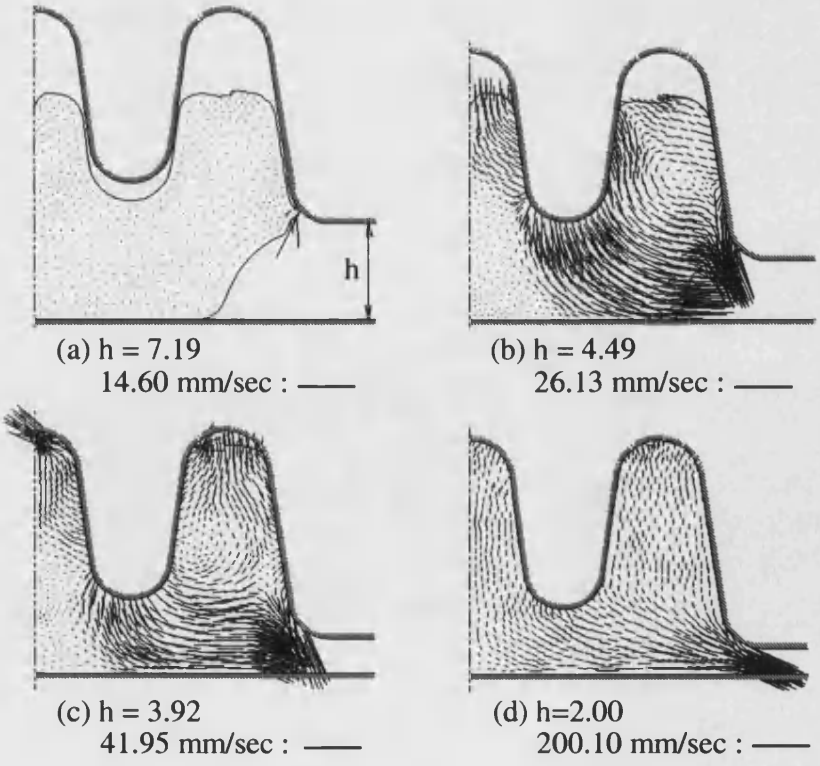


Figure 7.58: Velocity fields in the third stage forward simulation of Case Study 3.

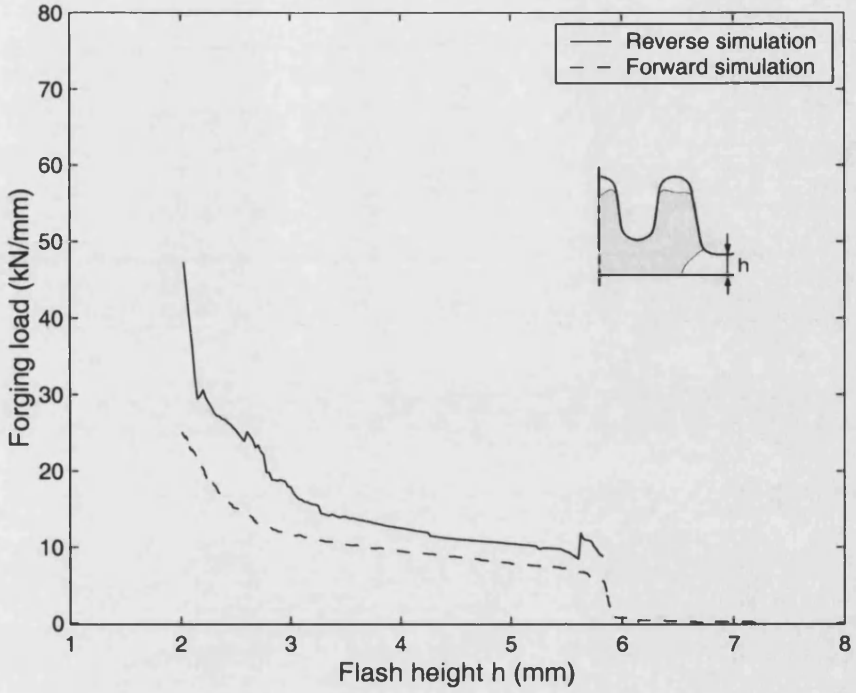


Figure 7.59: Forward and reverse load predictions for the first stage of Case Study 3.

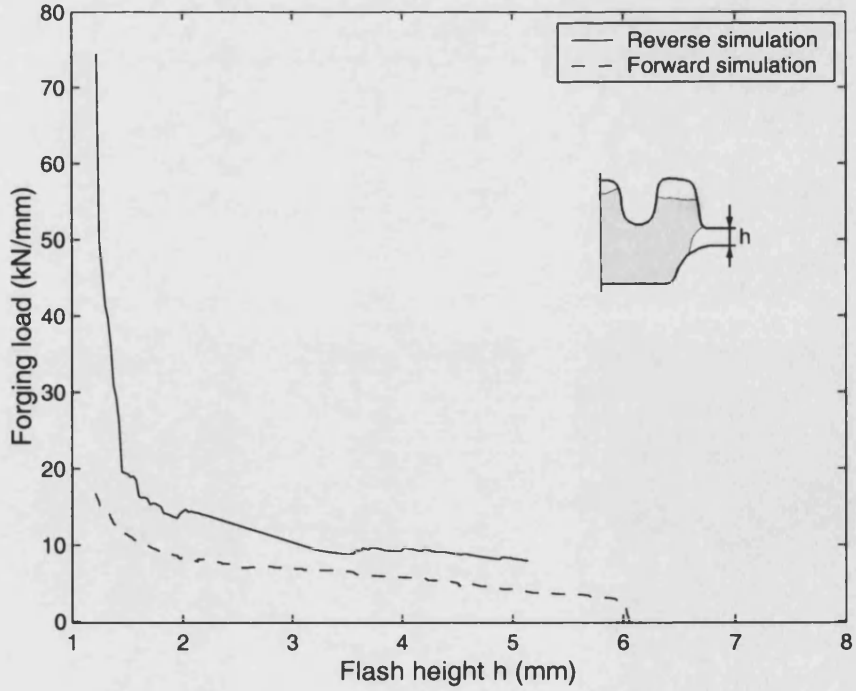


Figure 7.60: Forward and reverse load predictions for the second stage of Case Study 3.

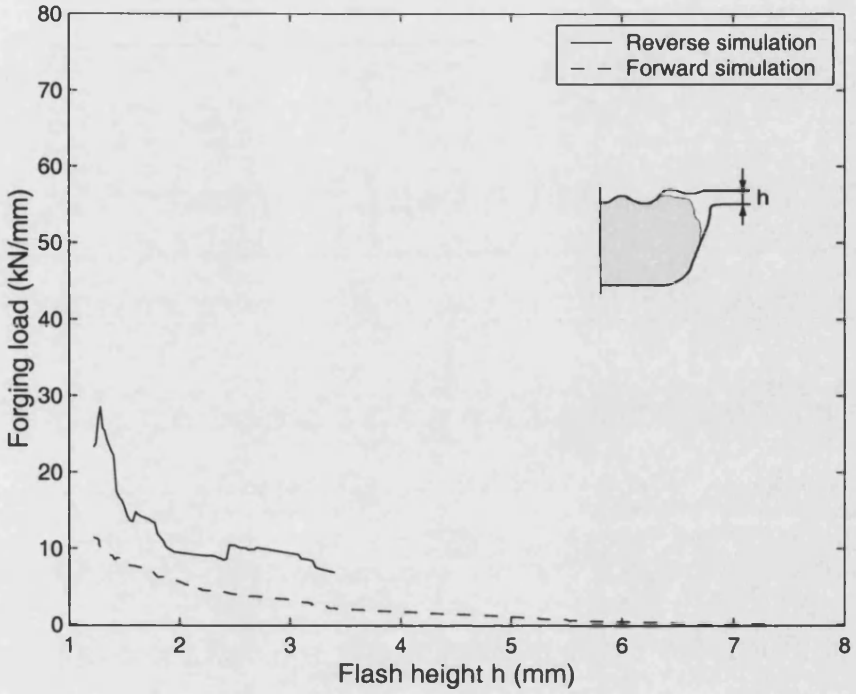


Figure 7.61: Forward and reverse load predictions for the third stage of Case Study 3.

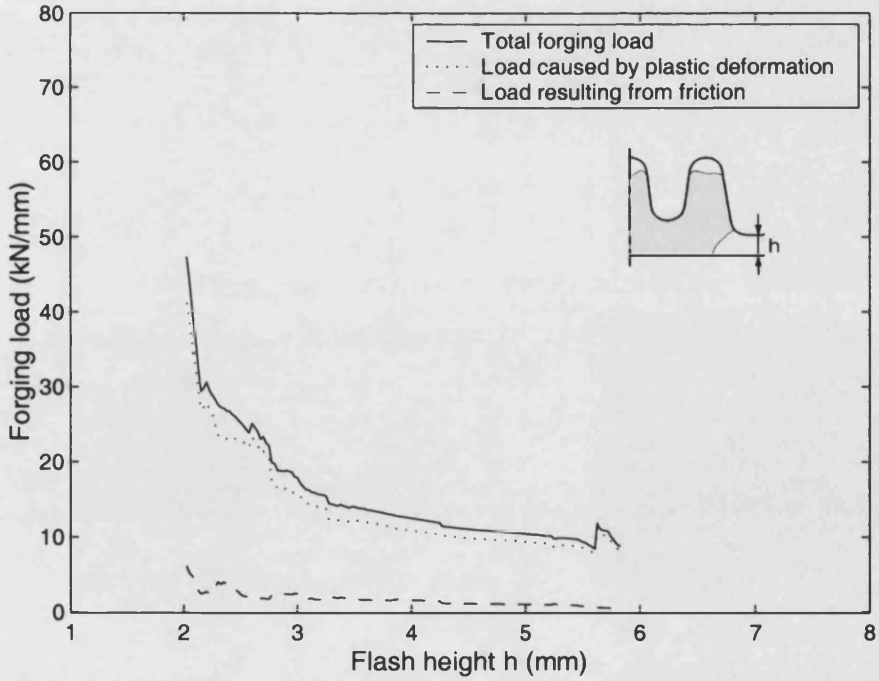


Figure 7.62: Load predictions for the first stage reverse simulation of Case Study 3.

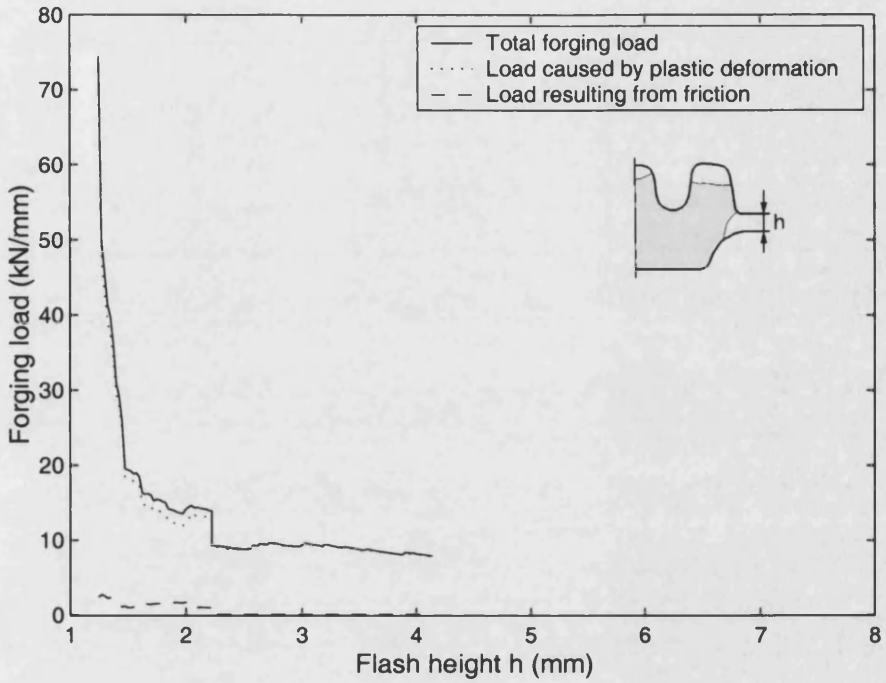


Figure 7.63: Load predictions for the second stage reverse simulation of Case Study 3.

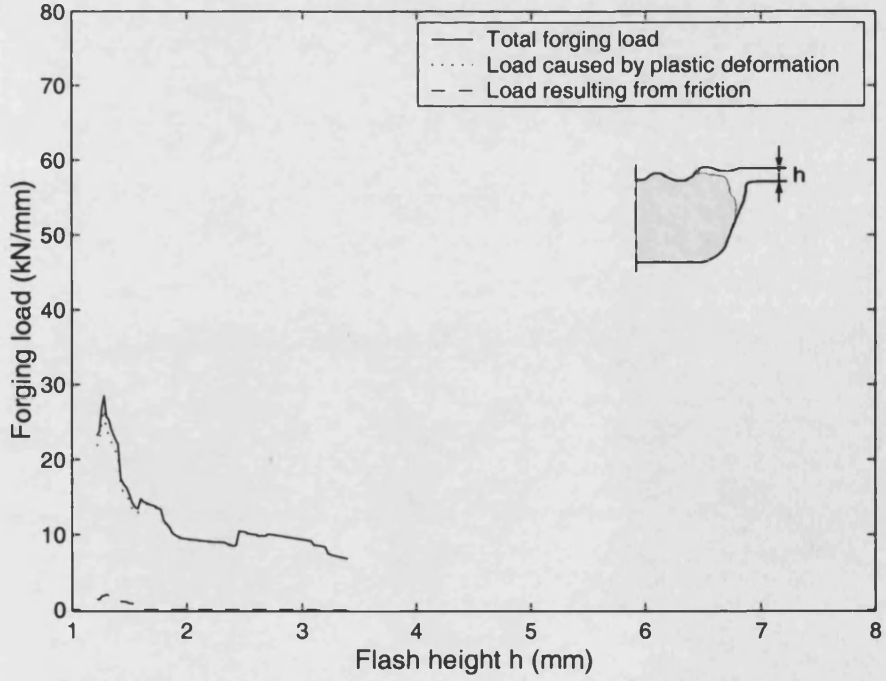
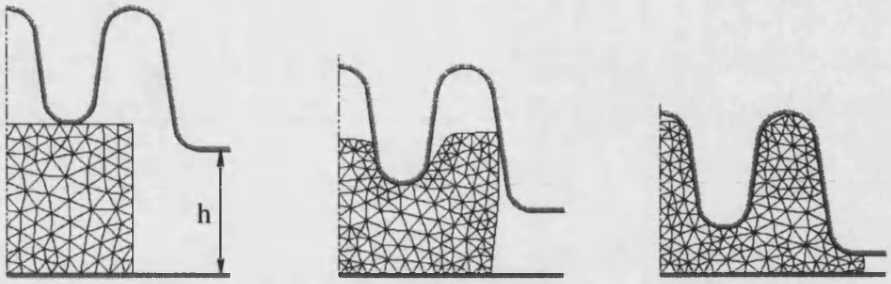
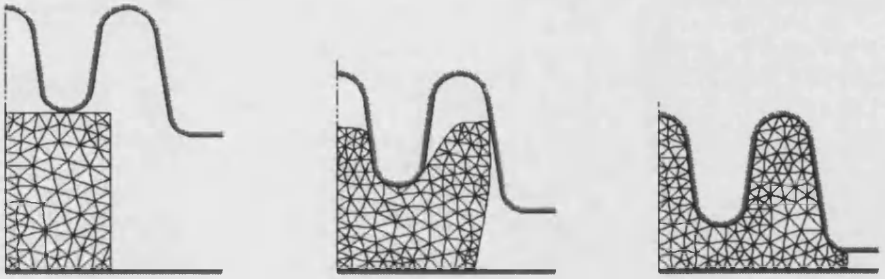


Figure 7.64: Load predictions for the third stage reverse simulation of Case Study 3.



Case 3A (a) h = 13.90 (b) h = 6.87 (c) h = 2.00
Billet size (width x height)
= 14.00 x 16.94 mm



Case 3B (d) h = 16.74 (e) h = 6.68 (f) h = 2.00
Billet size (width x height)
= 12.00 x 19.77 mm

Case 3C (g) h = 20.68 (h) h = 6.62 (i) h = 2.00
Billet size (width x height)
= 10.00 x 23.72 mm

Figure 7.65: Forward simulations using one-stage forging with different billet sizes for Case Study 3.

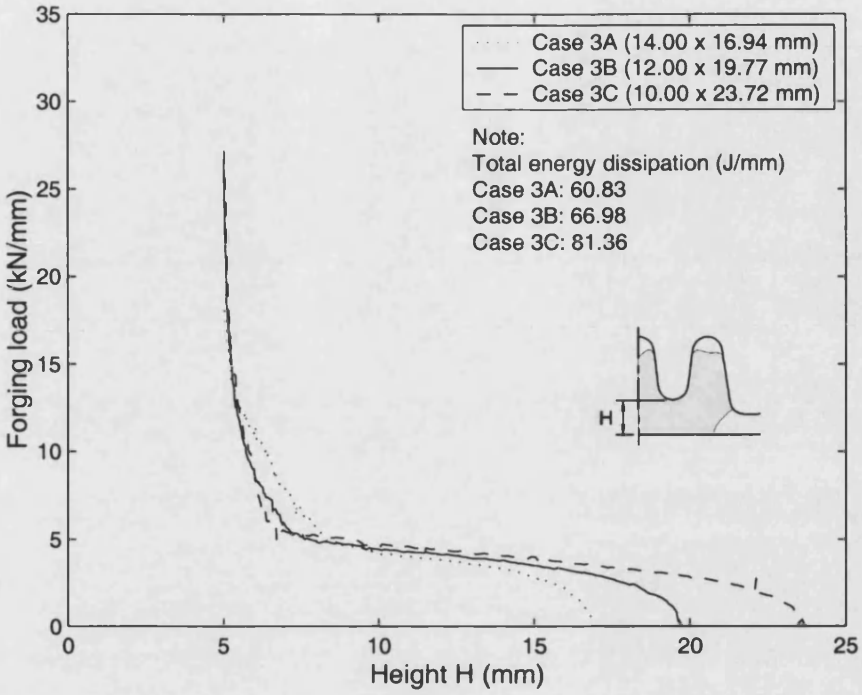


Figure 7.66: Forward load predictions for the one-stage forging simulation using different billet sizes for Case Study 3.

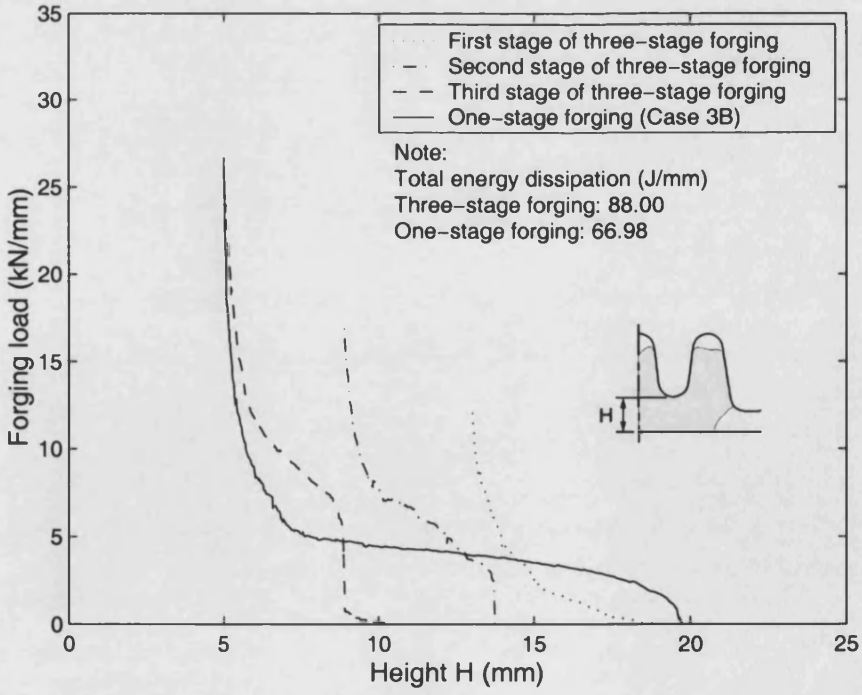
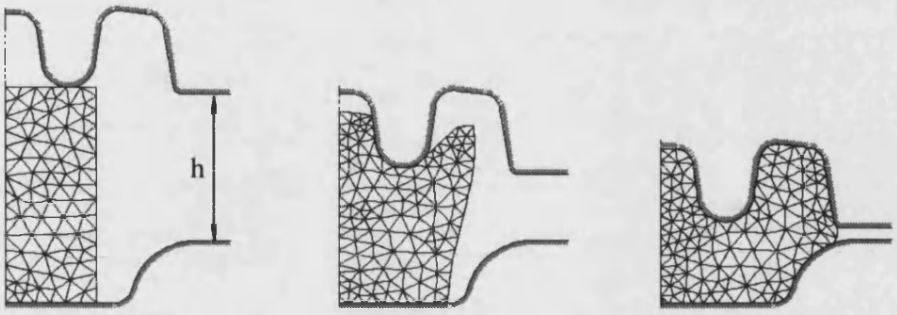


Figure 7.67: Load predictions for the one-stage and three-stage forward simulations of Case Study 3.

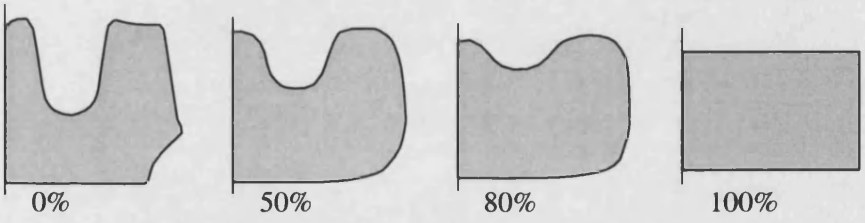


(a) h = 16.25
Billet size (width x height)
= 10.00 x 23.97mm

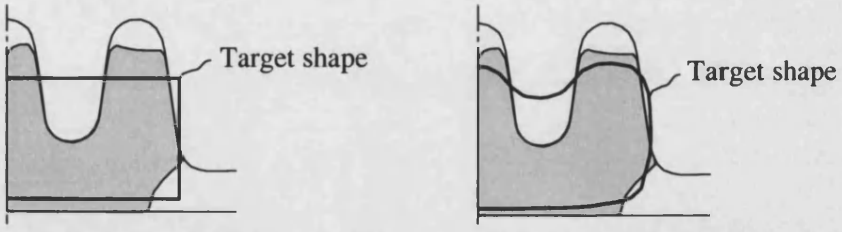
(b) h = 7.20

(c) h = 1.2

Figure 7.68: Forward simulation using a billet for the second stage preform of Case Study 3.



(a) Imagined metamorphism from the deformed workpiece to a rectangular target shape and the degree of the difference compared with the deformed workpiece



(b) Target shape determined by the simplest shape, i.e. the shape with 100% difference

(c) Target shape determined by the shape with 80% difference

Figure 7.69: Imagined metamorphism of the deformed workpiece and the shapes for determining the target shapes in the reverse simulation.

Chapter 8

Conclusions

A preform design approach for forging has been developed using reverse simulation starting from a finished shape to the billet or a preform in which multi-stage forging is needed. The simulation is based on the upper bound theory with the finite element formulation and solves a velocity field which represents the minimum dissipation energy of the forging process subject to the boundary conditions. The developed design approach has been evaluated by experiments and numerical simulations for two-dimensional plane strain cases. Assessment has shown a potential for the forging industry. Directions for future work have also been revealed, especially the determination of the target shape during the reverse simulation.

Modelling of Forging Process Using the Upper Bound Theorem with the FEM

The modelling of the forging process was continued from previous work which was carried out by Joe Pitt-Francis whose work was concentrated on forward simulation. The formulation of the upper bound with the finite element proce-

dure, including the friction and incompressibility terms, has been summarised and checked with the computer code of TEUBA. A new version of the user and developer manual is also being written, which includes the proposed reverse simulation approach for forging preform design.

A reducing time step technique has been invented and implemented into the code to improve the guess of the velocity field for the Newton-Raphson solver in the second and sequential increments. This technique reduces the difference between two increments and increases the possibility of success in solving the velocity field by using a previous solution to set an initial guess for the current increment.

It has been found that some velocity components of finite element nodes approach zero during the simulation. This situation causes the denominator of the system equations to approach zero and thus leads to a numerical problem. The problem is solved by assigning a small value to these velocity components and therefore the simulation can be continued smoothly afterwards.

A penalty factor which affects the incompressibility of the plastic deformation and the stability of the simulation has been investigated. An appropriate value has been selected for the simulation. Moreover, the effect of the number of elements on the modelling becomes more significant as the value of the friction factor increases.

According to the assessment carried out in this thesis and by Joe Pitt-Francis, the upper bound theory with the finite element is a rapid and approximate approach for modelling forging process. The modelling is also able to respond to the effect of friction in terms of a Tresca friction model with a constant factor.

Reverse Simulation for Preform Design

A preform design using the reverse simulation approach has been developed. The simulation employs the upper bound theory and the backwards tracing scheme to search a reverse path of the deformation from a finished product.

A measure of material distribution formulated by the second moment of area has been developed to set a rectangular shape which is assumed as the simplest and closest target shape to the reversed workpiece of two-dimensional plane strain cases. The contact region, which is located outside and has the furthest distance to the boundary of the target shape, is regarded as the most complex region and should be released first. New boundary conditions are then formed and a reversed shape is thus predicted. A measurement of complexity has been proposed to compare and select one from two shapes resulting from the current contact conditions and new formed contact conditions which are determined by the target shape and the detachment of the contact region. The reverse simulation finally reduces the complexity of the workpiece and is thus able to predict a preform or a shape close to a rectangle.

Experiments and numerical simulations including the reverse approach using TEUBA and the forward prediction using FORGE2 have been studied for the one-cavity forging. The material and the friction factor were also measured for the aluminium alloy BS6063. The results show that reverse simulation is able to predict a billet which corresponds to the preform for the one-stage forging process. By using the predicted billet, nearly complete die filling is achieved according to the products produced by the experiments and the results predicted by the forward simulation. The assessment has also shown that the trends of the deformation predicted by both reverse and forward simulations are generally similar to the experiments. Moreover, the predicted forging load is higher than

the load measured in the experiments. This situation is particularly obvious in the prediction of the reverse simulation. In comparison with the macrostructure of the experimental products, the simulation has shown the ability to predict a velocity field which is useful to understand the material flow during forging process.

Multi-stage forgings including the two-cavity and three-cavity products have been studied using numerical simulation. The proposed reverse simulations have predicted preforms and billets. These shapes were then evaluated by the forward simulations. The assessment has shown that the predicted preforms and billets can successfully achieve the requirement of complete die filling although some cavities of the preforming stages are not completely filled. The deformation predicted by the reverse simulation is generally in accord with that predicted by the forward simulation. Moreover, the reverse simulation predicts higher forging loads for most forming stages in comparison with those predicted by the forward simulation. The velocity field is also generally similar in both types of simulations of multi-stage forging.

Recommendation of Future Work

The formulation of the dissipation energy of plastic deformation is derived by the finite element procedure using triangle elements to linearly interpolate the velocity field between nodes. This linear formulation provides an approximate and rapid simulation for forging process. However, the numerical experiments have shown a non-linear formulation may provide a better solution with smoother boundary of the deformed workpiece as shown in the results predicted by FORGE2 which uses six-noded triangle elements. Moreover, triangle elements might not be a good choice to formulate the problem of incompressibility in which a technique of reducing integration points may be needed [129]. Four-noded rectangle or six-

noded triangle elements could be investigated in future studies of formulation.

The determination of the new boundary conditions is the main difficulty in running the reverse simulation for preform design. To solve this difficulty, this thesis has proposed a new concept of controlling the distribution of material to set the target shape and thus determine new boundary conditions. The difference between the target shape and the deforming workpiece might be too great and thus possibly affect the prediction of preforms. Image metamorphosis that creates a smooth transition between two images could be used to develop a more rigorous strategy for determining the target shape for reverse simulation.

The proposed concept of preform design is based on reducing the complexity of the workpiece by means of controlling the distribution of material, and is mainly investigated using plane strain forging cases. The extension of the proposed concept to axisymmetric and three-dimensional forging problems would be one of the future studies in the modelling of forging process, especially for preform design using a reverse simulation approach.

References

[1] Dallas, D.B. (1976). *Tool and Manufacturing Engineers Handbook*, pp. 14.1 - 14.77. 3rd ed. New York: McGraw-Hill Book Company.

[2] Smith, E.H. (1994). *Mechanical Engineer's Reference Book*, pp. 16/12 - 16/54. 12th ed. Oxford: Butterworth-Heinemann Ltd.

[3] Penton Publishing Inc. (1995). Forgings beat castings for critical components. *Forging*, **Summer**, pp. 103-104.

[4] Altan, T., Oh, S.I. and Gegel, H.L. (1983). *Metal Forming: Fundamentals and Applications.* Ohio, USA: American Society for Metals.

[5] Sheljaskvo, S. (1996). Warm forging — a technology for manufacturing of precision components. **In:***Advanced Technology of Plasticity 1996: Proceedings of the 5th International Conference on Technology of Plasticity*, Columbus, Ohio, USA, October 1996 (T. Altan ed.), **I**, pp. 485-490.

[6] DeGarmo, E.P., Black, J.T. and Kohser, R.A. (1988). *Materials and Processes in Manufacturing.* 7th ed. New York: Macmillan Publishing Company.

[7] Kalpakjian, S. (1992). *Manufacturing Engineering and Technology.* 2nd ed. New York: Addison-Wesley Publishing Company.

[8] Lange, K. (1985). *Handbook of Metal Forming.* New York: McGraw-Hill Book Company.

[9] Dieter, G.E. (1988). *Mechanical Metallurgy*, London: McGraw-Hill Book Company.

[10] Brucelle, O. and Bernhart, G. (1999). Methodology for service life increase of hot forging tools. *Journal of Materials Processing Technology*, **87**(1-3), pp. 237-246.

[11] Saigal, A., Zhen, K. and Chan, T.S. (1995). Forging of compressor blades: temperature and ram velocity effects. *Transaction of the ASME: Journal of Engineering for Gas Turbines and Power*, **117**, pp. 528-533.

[12] FORGE2. CEMEF, Sophia-Antipolis, France.

[13] DEFORM. Scientific Forming Technologies Corporation, 700 Ackerman Rd., Suite 255, Columbus Ohio 42302-1559, USA.

[14] QForm. Quantor Ltd. P.O.Box 39, 117049 Moscow, Russia.

[15] Vemuri, K.R., Oh, S.I. and Altan, T. (1989). BID: A knowledge-based system to automate blocker design. *International Journal of Machine Tools and Manufacture*, **29**(4), pp. 505-518.

[16] Bramley, A.N. (1987). Computer aided forging design. *Annals of CIRP*, **36/1/1987**, pp. 135-138.

[17] Forging Modelling Project Report FMP-31. (1997). *Final Project Report.* Forging Modelling Project in Department of Mechanical Engineering at University of Bath. British EPSRC Grant Ref: GR/J61473.

[18] Bramley, A.N. (1999). From Kudo to TEUBA. **In**: *Umformtechnik 2000 Plus*, Germany, Bamberg: Meisenbach GmbH. (Herausgeber M. Geiger ed.), pp. 91-99.

[19] Cermak, J. and Valasek, M. (1998). Knowledge-based selection of metal forming technology. **In**: *International Conference on Forging and Related Technology (ICFT'98)*, Birmingham, 27-28 April 1998, pp. 363-372. Bury St Edmunds and London: Professional Engineering Publishing Limited for the Institution of Mechanical Engineers.

[20] Penton Publishing Inc. (1994). Metals and alloys. *Forging*, **Spring**, pp. 31-38.

[21] Penton Publishing Inc. (1994). Die lubricants. *Forging*, **Spring**, pp. 39-44.

[22] American Society for Metals. (1978). Forging and Forging Design. **In**: *Source Book on Slecetion and Fabrication of Aluminum Alloys*, Section V, pp. 85-147. Ohio, USA: American Society for Metals.

[23] Mynors, D.J., Bramley, A.N. and Allen, M. (1997). An examination of manual die design procedures as a precursor to the application of simulation.

In: *International Conference and Exhibition on Design and Production of Dies and Molds*, Turkey, 19-21 June 1997 (M.Akkok et al. ed.), pp. 283-290.

[24] Forging Modelling Project Report FMP-04. (1996). *Forging die design methods and the potential for modelling.* Forging Modelling Project in Department of Mechanical Engineering at University of Bath. British EPSRC Grant Ref: GR/J61473.

[25] ASM Handbook Committee. (1978). Properties and Selection: Irons and Steels. **In**: *Metals Handbook*, Volume 1. 9th ed., pp. 349-375. Ohio: American Society for Metals.

[26] American Society for Metals. (1972). *Forging Design Handbook.* Ohio: American Society for Metals.

[27] Byrer, T.G. ed., Semiatin, S.L. and Vollmer, D.C. eds. (1985). *Forging Handbook.* Ohio: Forging Industry Association and American Society for Metals.

[28] Kalpakjian, S. (1997). *Manufacturing Processes for Engineering Materials*, pp. 319-321. Harlow, England: Addison-Wesley Publishing Company.

[29] Schey, J.A. (1987). *Introduction to Manufacturing Processes*, pp. 238-241. New York: McGraw-Hill Book Company.

[30] Cho, N.C., Lee, S.H., Hong, S.S. and Lee, J.S. (1994). Experimental-verification of preform design for axisymmetrical heavy forging on a model hammer. *Journal of Materials Processing Technology*, **47**(1-2), pp. 103-110.

[31] Lee, R.S. and Hsu, Q.C. (1992). Development of an integrated process planning-based CAE system for cold forging. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, **206**(3), pp. 215-225.

[32] Ward, M.J., Miller, B.C. and Davey, K. (1998). Simulation of a multi-stage railway wheel and tyre forming process. *Journal of Materials Processing Technology*, **80-81**, pp. 206-212.

[33] Qu, H. and Balendra, R. (1998). Preform design for forging of aerofoil sections using FE simulation. *Journal of Materials Processing Technology*, **80-81**, pp. 144-148.

[34] Qu, H. and Balendra, R. (1998). Die-elasticity for precision forging of aerofoil sections using finite element simulation. *Journal of Materials Processing Technology*, **76**(1-3), pp. 56-61.

[35] Tsoukalas, L.H. and Uhring, R.E. (1997). *Fuzzy and Neural Approaches in Engineering.* New York: John Wiley and Sons, Inc.

[36] Rembold, U., Nnaji, B.O. and Storr, A. (1993). *Computer Integrated Manufacturing and Engineering*, pp. 135-175. Reading, England: Addison-Wesley Publishing Company.

[37] Bakhshi-Jooybari, M., Pillinger, I., Dean, T.A. and Hartley, P. (1996). Development of product and process comparison criteria for an intelligent knowledge-based system for forging die design. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, **210**, pp. 565-578.

[38] Lee, R.S., Hsu, Q.C. and Sheu, J.J. (1998). Development of computer integrated manufacturing system for cold forging. **In:** *International Conference on Forging and Related Technology (ICFT'98)*, Birmingham, 27-28 April 1998, pp. 373-382. Bury St Edmunds and London: Professional Engineering Publishing Limited for the Institution of Mechanical Engineers.

[39] Hinton, G.E. (1992). How neuarl networks learn from experience. *Scientific American*, **267**(3), pp. 105-109.

[40] Roy, R., Chodnikiewicz, K. and Balendra, R. (1994). Interpolation of forging preform shapes using neural networks. *Journal of Materials Processing Technology*, **45**(1-4), pp. 695-702.

[41] Kim, D.J., Kim, B.M. and Choi, J.C. (1997). Determination of the initial billet geometry for a forged product using neural networks. *Journal of Materials Processing Technology*, **72**(1), pp. 86-93.

[42] Hsu, Q.C. and Lee, R.S. (1997). Cold forging process design based on the induction of analytical knowledge. *Journal of Materials Processing Technology*, **69**(1-3), pp. 264-272.

[43] Rebizant, W. (1999). Artificial Intelligence Methods — Fundamentals: Fuzzy Logic, Expert Systems, Artificial Neural Networks. **In:** *Series of Presentations* on Thursday 13 May 1999 in the Department of Electronic and Electrical Engineering at the University of Bath.

[44] Holland, J.H. (1992). Genetic algorithms. *Scientific American*, **267**(1), pp. 44-50.

[45] Davis, L. ed. (1991). *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhod.

[46] Roy, S., Ghosh, S. and Shivpuri, R. (1997). A new approach to optimal design of multi-stage metal forming processes with micro genetic algorithms. *International Journal of Machine Tools and Manufacture*, **37**(1), pp. 29-44.

[47] Chung, J.S., and Hwang, S.M. (1998). Application of a genetic algorithm to precess optimal design in non-isothermal metal forming. *Journal of Materials Processing Technology*, **80-81**, pp. 136-143.

[48] Fourment, L. and Chenot, J.L. (1996). Optimal design for non-steady-state metal forming processes – I. Shape optimization method. *International Journal for Numerical Methods in Engineering*, **39**, pp. 33-50.

[49] Fourment, L., Balan, T. and Chenot, J.L. (1996). Optimal design for non-steady-state metal forming processes – II. Application of shape optimization in forging. *International Journal for Numerical Methods in Engineering*, **39**, pp. 51-65.

[50] Zhao, G., Wright, E. and Grandhi, R.V. (1997). Preform die shape design in metal forming using an optimization method. *International Journal for Numerical Methods in Engineering*, **40**, pp. 1213-1230.

[51] Zhao, G., Wright, E. and Grandhi, R.V. (1997). Sensitivity analysis based preform die shape design for net-shape forging. *International Journal of Machine Tools and Manufacture*, **37**(9), pp. 1251-1271.

[52] Zhao, G., Huff, R., Hutter, A. and Grandhi, R.V. (1997). Sensitivity analysis based preform die shape design using the finite element method. *Journal of Materials Engineering and Performance*, **6**(3), pp. 303-310.

[53] Kusiak, J. (1996). A technique of tool-shape optimization in large scale problems of metal forming. *Journal of Materials Processing Technology*, **57**, pp. 79-84.

[54] Lapovok, R. and Thomson, P.F. (1997). An approach to the optimal design of rolling passes. *International Journal of Machine Tools and Manufacture*, **37**(8), pp. 1143-1154.

[55] Lapovok, R. (1998). Improvement of die life by minimisation of damage accumulation and optimisation of preform design. *Journal of Materials Processing Technology*, **80-81**, pp. 608-612.

[56] Park, J.J. and Rebelo, N. and Kobayashi, S. (1983). A new approach to preform design in metal forming with the finite element method. *International Journal of Machine Tools and Manufacture*, **23**(1), pp. 71-79.

[57] Hwang, S.M. and Kobayashi, S. (1984). Preform design in plane strain rolling by the finite element method. *International Journal of Machine Tool Design and Research*, **24**(4), pp. 253-266.

[58] Kang, B.S., Kim, N. and Kobayashi, S. (1990). Computer-aided preform design in forging of an airfoil section blade. *International Journal of Machine Tools and Manufacture*, **30**(1), pp. 43-52.

[59] Kang, B.S., Lee, J.H, and Kim S.Y. (1997). Development of a methodology to form net-shape nosing shells by the backwards tracing scheme of the rigid-plastic. *International Journal of Machine Tools and Manufacture*, **37**(6), pp. 737-750.

[60] Kang, B.S., Kim, B.M. and Choi, J.C. (1994). Prefrom design in extrusion by the FEM and its experimental confirmation. *Journal of Materials Processing Technology*, **41**, pp. 237-248.

[61] Kang, B.S., Lee, J.H. and Choi, H.H. (1997). Extension of the backward tracing scheme of the rigid-plastic FEM in three dimensional deformation. *Transaction of the ASME: Journal of Manufacturing Science and Engineerin*, **119**, pp. 556-562.

[62] Han, C.S., Grandhi, R.V. and Srinivasan, R. (1993). Optimum design of forging die shapes using nonlinear finite element analysis. *AIAA Journal*, **31**(4), pp. 774-781.

[63] Srinivasan, R., Reddy, G.H.K., Kumar, S.S., and Grandhi, R.V. (1994). Intermediate shapes in closed-die forging by the backward deformation optimisation method (BDOM). *Journal of Materials Engineering and Performance*, **3**(4), pp. 501-513.

[64] Zimmermann, H.-J. (1994). *Fuzzy set theory and Its Applications.* 2nd. ed. Boston: Kluwer Academic Publishers.

[65] Biglari, F.R., O'Dowd, N.P. and Fenner, R.T. (1998). Optimum design of forging dies using fuzzy logic in conjunction with the backward deformation method. *International Journal of Machine Tools and Manufacture*, **38**(8), pp. 981-1000.

[66] Zhao, G., Wright, E. and Grandhi, R.V. (1996). Computer aided preform design in forging using the inverse die contact tracking method. *International Journal of Machine Tools and Manufacture*, **36**(7), pp. 755-769.

[67] Zhao, G., Wright, E. and Grandhi, R.V. (1996). Preform design in bulk forging processes. **In**: *Advanced Technology of Plasticity 1996: Proceedings of the 5th International Conference on Technology of Plasticity*, Columbus, Ohio, USA, October 1996 (T. Altan ed.), Vol. I, pp. 453-456.

[68] Zhao, G., Wright, E. and Grandhi, R.V. (1995). Forging preform design with shape complexity control in simulating backward deformation. *International Journal of Machine Tools and Manufacture*, **35**(9), pp. 1225-1239.

[69] McDermott, R.P. and Bramley, A.N. (1974). Forging analysis — a new approach. *Metallurgia and Metal Forming.* **May**, pp. 127-130.

[70] Kim, H.Y. and Kim, D.W. (1994). Computer-aided preform design in the closed-die forging process. *Journal of Materials Processing Technology*, **41**(1), pp. 83-104.

[71] Liu, Q.B., Wu, S.C. and Sun, S. (1998). Preform design in axisymmetric forging by a new FEM-UBET method. *Journal of Materials Processing Technology*, **74**(1-3), pp. 218-222.

[72] Hill, R. (1950, 1985). *The Mathematical Theory of Plasticity*, pp. 49-69. Oxford: Oxford University Press.

[73] Johnson, W. and Mellor, P.B. (1973). *Engineering Plasticity*, pp. 415-419. London: Van Nostrand Reinhold Company.

[74] Nielsen, M.P. (1984). *Limit Analysis and Concrete Plasticity*, pp. 9-12. New Jesery: Prentice-Hall, Inc.

[75] Slater, R.A.C. (1977). *Engineering Plasticity: Theory and Application to Metal Forming Processes*, p. 5; pp. 79-82; pp. 356-359. New York: John Wiley & Sons Inc.

[76] Kudo, H. (1960). An upper-bound approach to plane-strain forging and extrusion—I. *International Journal of Mechanical Sciences*, **1**, pp. 57-83.

[77] Kudo, H. (1960). Some analytical and experimental studies of axisymmetric cold forging and extrusion—I. *International Journal of Mechanical Sciences*, **2**, pp. 102-127.

[78] Kobayashi, S. (1964). Upper-bound solutions of axisymmetric forming problems—I. *Transactions of the ASME: Journal of Engineering for Industry*, **86**, pp. 122-126.

[79] Kobayashi, S. (1964). Upper-bound solutions of axisymmetric forming problems—II. *Transactions of the ASME: Journal of Engineering for Industry*, **86**, pp. 326-332.

[80] Osman, F.H. (1981). *Computerized Simulation of Forging Processes.* Ph.D. Thesis, University of Leeds.

[81] Oudin, J. and Ravalard, Y. (1981). An upper bound method for computing loads and flow patterns in plane-strain forging process. *International Journal of Machine Tools and Manufacture*, **21**(3-4), pp. 237-250.

[82] Ghobrial, M.I. (1985). *Computer Aided Analysis for Axisymmetric Forging.* PhD. Thesis, University of Leeds.

[83] Lugora, C.F. and Bramley, A.N. (1989). Three-dimensional analysis of closed-die forging processes, Part 1: formulation. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, **203**, pp. 33-37.

[84] Lugora, C.F. and Bramley, A.N. (1989). Three-dimensional analysis of closed-die forging processes, Part 2: applications. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, **203**, pp. 38-42.

[85] Lee, R.S. and Kwan, C.T. (1996). A three-dimensional upper bound elemental technique for forging analysis. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, **210**, pp. 353-359.

[86] Liu, Y.H., Cen, Z.Z. and Xu, B.Y. (1995). A numerical method for plastic limit analysis of 3-D structures. *International Journal of Solids and Structures*, **32**(12), pp. 1645-1658.

[87] Pitt-Francis, J.M., Bowyer, A. and and Bramley, A.N. (1996). TEUBA — a simple 3D formulation for modelling forging using the upper bound method. *Annals of the CIPR*, **45/1/1996**, pp. 245-248.

[88] Hwan, C.L. (1997). Plane strain extrusion by sequential limit analysis. *International Journal of Mechanical Sciences*, **39**(7), pp. 807-817.

[89] Hwan, C.L. (1997). An upper bound finite element procedure for solving large plane strain deformation. *International Journal for Numerical Methods in Engineering*, **40**, pp. 1909-1922.

[90] Kobayashi, S., Oh S.I., and Altan, T. (1989). *Metal Forming and the Finite-Element Method.* Oxford: Oxford University Press.

[91] McMeeking, R.M. and Rice, J.R. (1975). Finite-element formulations for problems of large elastic-plastic deformation. *Int. J. Solids Structures*, **11**, pp. 601-616.

[92] Nagtegaal, J.C. and Veldpaus, F.E. (1984). On the implementation of finite strain plasticity equations in a numerical model. **In:** *Numerical Analysis of Forming Processes* (J.F.T. Pittman, O.C. Zienkiewicz, R.D. Wood and J.M. Alexander eds.), pp. 351-371. Chichester: John Wiley & Sons Ltd.

[93] Zienkiewicz, O.C. and Taylor, R.L. (1994). *The Finite Element Method, Volume I*, 4th. ed, pp. 249-253. London: McGraw-Hill Book Company.

[94] Shimazaki, Y. and Thompson, E.G. (1981). Elasto visco-plastic flow with special attention to boundary conditions. *International Journal for Numerical Methods in Engineering*, **17**, pp. 97-112.

[95] Cheng, W.F. (1991). *Plasticity for Structural Engineers*, pp. 409-419. London: Springer-Verlag.

[96] Wagoner, R.H. and Chenot, J. (1997). *Fundamentals of Metal Forming*, pp. 263-267; p. 317. New York: John Wiley & Sons Inc.

[97] Edwards, L. and Endean M. (1995). *Manufacturing with Materials*, pp. 164-165. London: Butterworth-Heinemann Ltd.

[98] Chenot, J.L. and Bellet M. (1992). The viscoplastic approach for the finite element modelling of metal-forming process. **In:** *Numerical Modelling of Material Deformation Processes* (P. Hartley, I. Pillinger and C. Sturgess eds.), pp. 179-224. London: Spring-Verlag.

[99] Zienkiewicz, O.C. and Godbole, P.N. (1975). A penalty function approach to problems of plastic flow of metals with large surface deformations. *Journal of Strain Analysis*, **10**(3), pp. 180-183.

[100] Pitt-Francis, J.M. (1997). *TEUBA: User and Developer Documentation.* Department of Mechanical Engineering, University of Bath.

[101] Chang, C.C. and Pitt-Francis, J.M. (2000). *TEUBA: User and Developer Documentation.* Department of Mechanical Engineering, University of Bath.

[102] Reddy, J.N. (1986). *Applied Functional Analysis and Variational Methods in Engineering*, p. 460. London: McGraw-Hill Book Company.

[103] Golub, G.H. and Van Loan, C.F. (1996). *Matrix Computations*, 3rd ed. Baltimore: The Johns Hopkins University Press.

[104] Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992). *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge: Cambridge University Press.

[105] Forging Modelling Project Report FMP-17. (1996). *TEUBA: Performance assessment.* Forging Modelling Project in Department of Mechanical Engineering at University of Bath. British EPSRC Grant Ref: GR/J61473.

[106] Pascoe, S.K. (1997). Finite elements in contact problems. Lecture note: *Nonlinear Finite Element Analysis of Geometric Nonlinearities and Contact*, University of Nottingham, 3-4 July 1997.

[107] Luenberger, D.G. (1984). *Linear and Nonlinear Programming.* Reading, Massachusetts: Addison-Wesley Publishing Company.

[108] Pittman, J.F.T., Zienkiewicz, O.C., Wood, R.D. and Alexander, J.M. eds. (1984). *Numerical Analysis of Forming Processes.* Chichester: John Wiley & Sons Ltd.

[109] Lam, Y.C. and Zhang, X.Q. (1994). Evaluation of relaxation schemes for Newton-Raphson iteration in rigid-plastic finite-element analysis. *Journal of Materials Processing Technology*, **41**(4), pp. 361-373.

[110] Jackson, J.E. Jr and Ramesh M.S. (1992). The rigid-plastic finite-element method for simulation of deformation processing. **In**: *Numerical Modelling of Material Deformation Processes* (P. Hartley, I. Pillinger and C. Sturgess eds.), pp. 148-177. London: Spring-Verlag.

[111] Mori, K. (1992). Extrusion and drawing. **In:** *Numerical Modelling of Material Deformation Processes* (P. Hartley, I. Pillinger and C. Sturgess eds.), pp. 303-317. London: Spring-Verlag.

[112] CADfix. FEGS Ltd., Oakington House, Oakington, Cambridge, CB4 5AF, England.

[113] Tcl/Tk, Tcl 7.4 and Tk 4.0. Tcl/Tk Services & Packages, NeoSoft Inc., USA.

[114] Motif. The Open Group. Apex Plaza, Forbury Road, Reading, Berkshire, RG1 1AX, UK.

[115] OpenGL Architecture Review Board, Neider, J., Davis, T, and Woo, M. (1993). *OpenGL Programming Guide: The Offical Guide to Learning OpenGL, Release 1.* New York: Addison-Wesley Publishing Company.

[116] Wernecke, J. and Open Inventor Architecture Group. (1994). *The Open Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2.* New York: Addison-Wesley Publishing Company.

[117] Male, A.T. and Cockcroft, M.G. (1964). A method for determination of the coefficient of friction of metals under conditions of bulk plastic deformation. *Journal of the Institute of Metals.*, **93**, pp. 38-46.

[118] Hawkyard, J.B. and Johnson, W. (1967). An analysis of the changes in geometry of a short hollow cylinder during axial compression. *International Journal of Mechanical Sciences*, **9**, pp. 163-182.

[119] *Operation and Maintenance Manual for 100 Ton Hydraulic CNC Press.* Joseph Rhodes Limited, Belle Vue, Wakefield, WF1 5FQ, England.

[120] ASTM. (1994). Standard Test Methods of Compression Testing of Metallic Materials at Room Temperature. *1994 Annual Book of ASTM Standards*, Section 3: Metals Test Methods and Analytical Procedures, Vol. 03.01, pp. 101-108.

[121] ROCOL MTS 1000 Bearing Grease. ROCOL Ltd., Leeds, England.

[122] SPSS Inc., 233 S. Wacker Drive, 11th Floor, Chicago, IL 60606-6307.

[123] American Society for Metals. (1995). *ASM Handbook, Volume 1: Properties and Selection: Irons, Steels, and High-Performance Alloys*, 4th ed., p.441, USA: ASM international.

[124] AutoCAD Release 14. Autodesk Inc., 111 McInnis Parkway, San Rafael, California 94903, USA.

[125] Chen, S.E. and Parent, R.E. (1989). Shape averaging and its applications to industrial design. *IEEE Computer Graphics and Applications.* **9**(1), pp. 47-54.

[126] Cheung, K.K., Yu, K.M. and Hui, K.C. (1997). Simulation of metal forming processes by volume-conserved metamorphosis. *Journal of Materials Processing Technology*, **63**, pp. 701-706.

[127] Lee, S.Y., Chwa, K.Y., Hahn, J. and Shin, S.Y. (1996). Image morphing using deformation techniques. *The Journal of Visualization and Computer Animation*, **7**, pp. 3-23.

[128] Savchenko, V. and Pasko, A. (1998). Transformation of functionally defined shapes by extended space mappings. *The Visual Computer*, **14**, pp. 257-270.

[129] Nagtegaal, J.C., Parks, D.M. and Rice, J.R. (9174). On numerically accurate finite element solutions in the fully plastic range. *Computer Methods in Applied Mechanics and Engineering*, **4**, pp. 153-177.

[130] Pitt-Francis, J.M. (1995). *TEUBA: Status Paper.* Forging Modelling Project in Department of Mechanical Engineering at University of Bath. British EPSRC Grant Ref: GR/J61473.

Appendix

A-1 Derivatives of Shape Functions

The derivatives of the linear shape functions for a three-noded element can be calculated by a matrix operation [130] described as follows.

Consider a triangle element which has three nodes with coordinates (x_1^k, x_2^k), where (x_1^k, x_2^k) are the coordinates of node k $(1 \leq k \leq 3)$. The shape function ϕ^j $(1 \leq j \leq 3)$ is

$$\phi^j = a_0^j + a_1^j x_1 + a_2^j x_2 \tag{A-1}$$

where a_0^j, a_1^j and a_2^j are constants. The function ϕ^j has the value 1 at j-th node and zero at the other nodes. The equations for shape function ϕ^1 can be expressed in the matrix form as

$$\begin{bmatrix} x_1^1 & x_2^1 & 1 \\ x_1^2 & x_2^2 & 1 \\ x_1^3 & x_2^3 & 1 \end{bmatrix} \begin{Bmatrix} a_1^1 \\ a_2^1 \\ a_0^1 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix} \tag{A-2}$$

Therefore, the equations for all shape functions in a triangle element are

$$\begin{bmatrix} x_1^1 & x_2^1 & 1 \\ x_1^2 & x_2^2 & 1 \\ x_1^3 & x_2^3 & 1 \end{bmatrix} \begin{bmatrix} a_1^1 & a_1^2 & a_1^3 \\ a_2^1 & a_2^2 & a_2^3 \\ a_0^1 & a_0^2 & a_0^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A-3}$$

Let the matrix **A** be defined by

$$\mathbf{A} = \begin{bmatrix} x_1^1 & x_2^1 & 1 \\ x_1^2 & x_2^2 & 1 \\ x_1^3 & x_2^3 & 1 \end{bmatrix} \tag{A-4}$$

and

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I} \tag{A-5}$$

Therefore, the inverse of the matrix is

$$\mathbf{A}^{-1} = \begin{bmatrix} a_1^1 & a_1^2 & a_1^3 \\ a_2^1 & a_2^2 & a_2^3 \\ a_0^1 & a_0^2 & a_0^3 \end{bmatrix} \tag{A-6}$$

The derivatives of shape functions in x_1 and x_2 directions at each node can be written in the matrix form as

$$\begin{bmatrix} \frac{\partial \phi^1}{\partial x_1} & \frac{\partial \phi^2}{\partial x_1} & \frac{\partial \phi^3}{\partial x_1} \\ \frac{\partial \phi^1}{\partial x_2} & \frac{\partial \phi^2}{\partial x_2} & \frac{\partial \phi^3}{\partial x_2} \\ . & . & . \end{bmatrix} = \begin{bmatrix} a_1^1 & a_1^2 & a_1^3 \\ a_2^1 & a_2^2 & a_2^3 \\ a_0^1 & a_0^2 & a_0^3 \end{bmatrix} \tag{A-7}$$

Finally, from equations (A-4), (A-5) and (A-7), it can be found that $\mathbf{A^{-1}}$ is the derivatives of shape function ϕ^k with respect to x_i $(1 \leq i \leq 2)$.

A-2 Tool Details for Experiments

Tool details for the one-stage forging experiment are given in Figures A.1, A.2 and A.3.

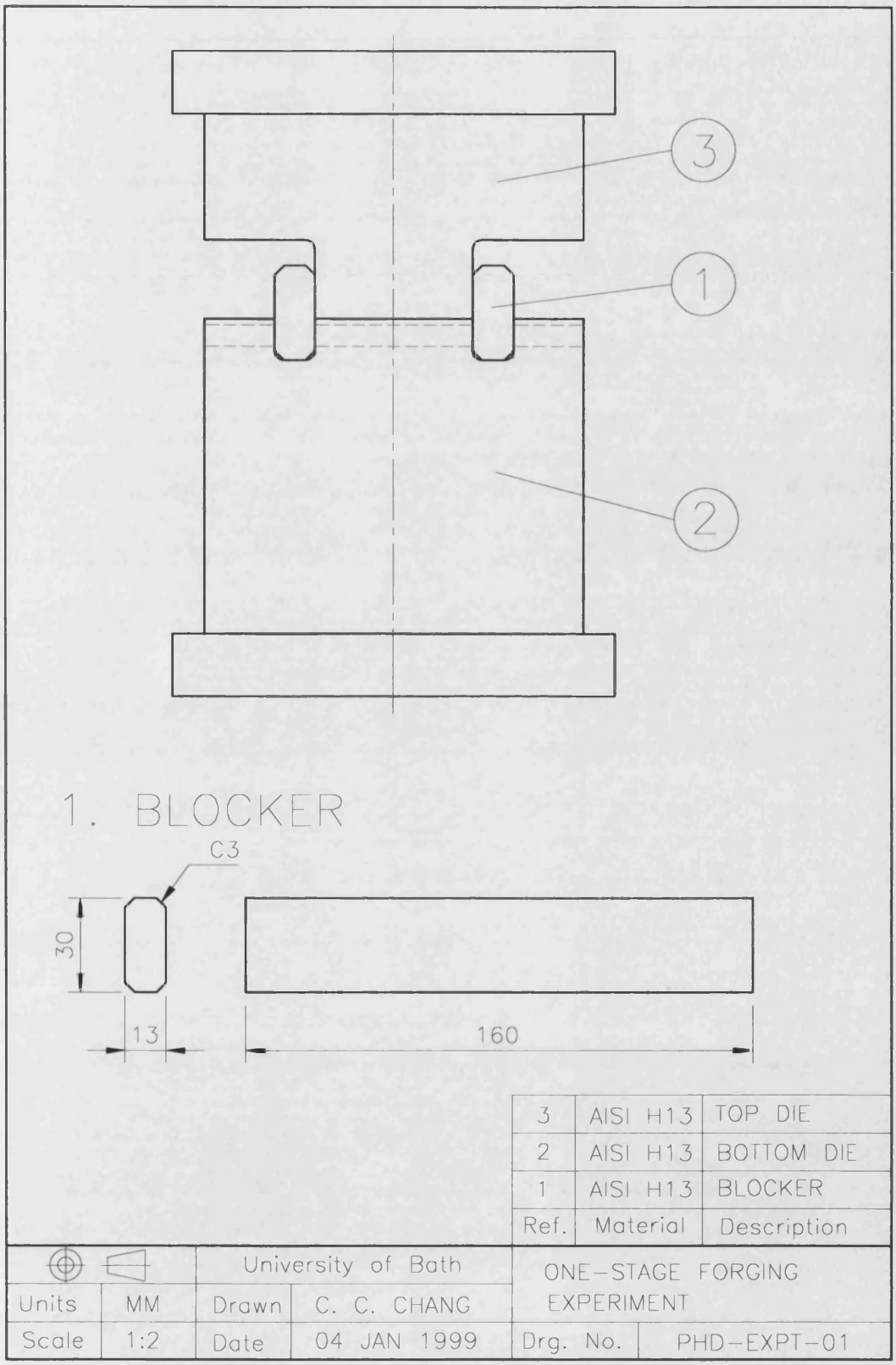


Figure A.1: Details of dies — Assembly and blocker.

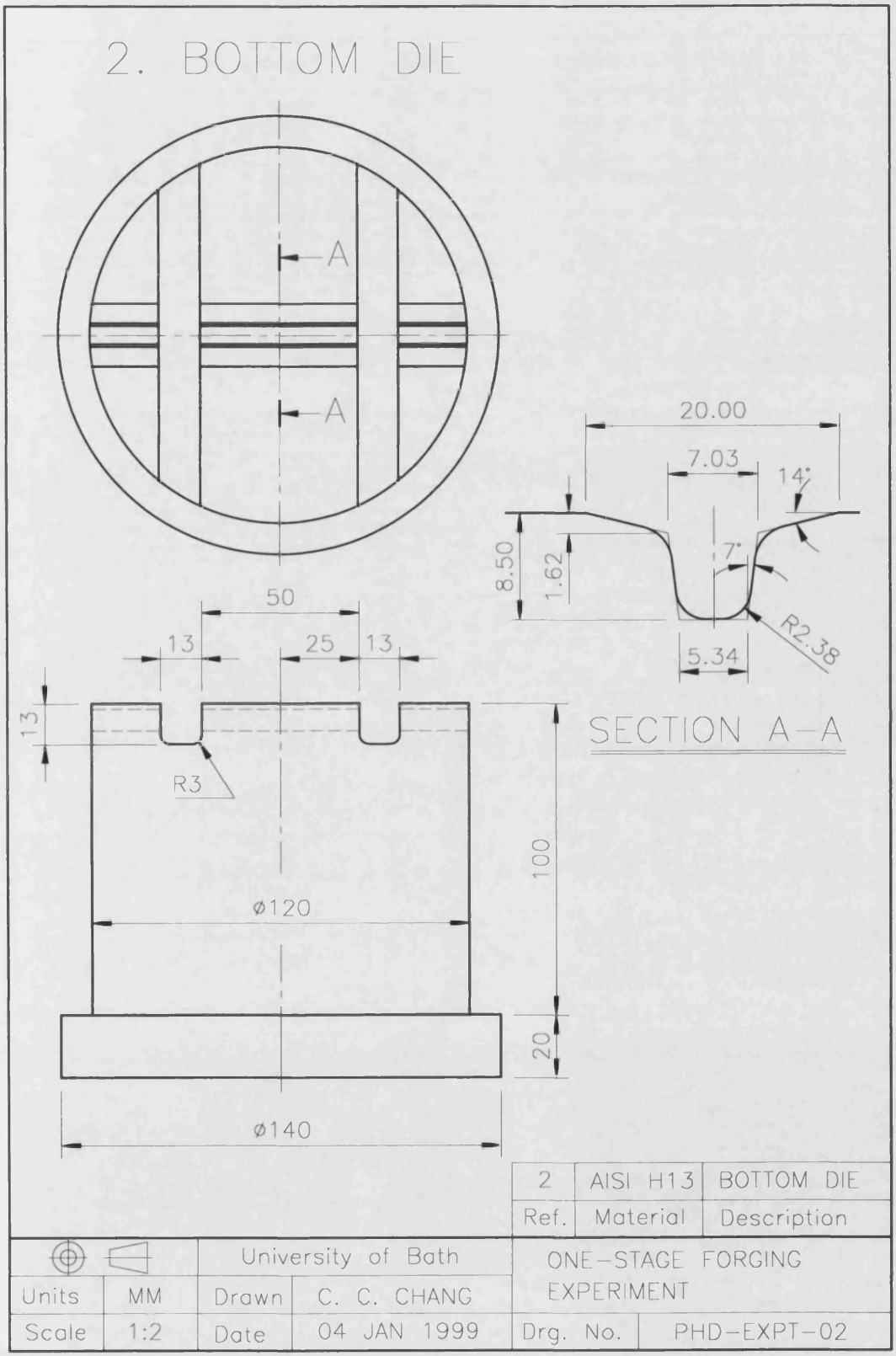


Figure A.2: Details of bottom die.

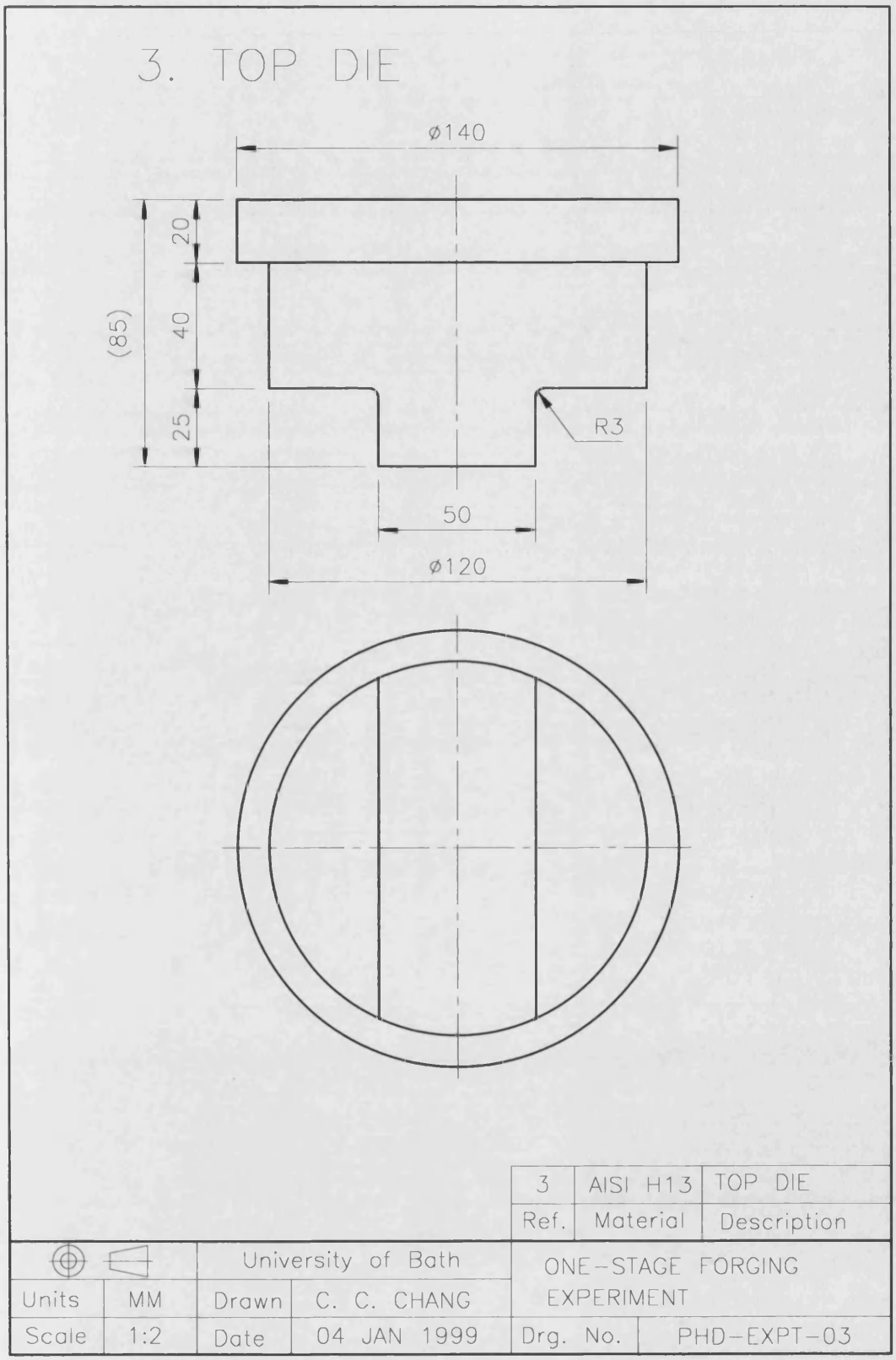


Figure A.3: Details of top die.

A-3 Key Computer Routines Developed and Implemented into TEUBA

A-3.1 Key Routines for Reverse Simulation

```
/*************************************************************
 * Reverse Tracing Scheme
 *************************************************************/
void
solve_reverse_tracing_scheme(struct mesh *a_mesh)
{
    struct node *a_node=NULL;
    int normal_result_case_1;
    int normal_result_case_2;
    int node_to_be_released=0;
    real best_function;
    printf("\tThere are currently %i nodes attached.\n", a_mesh->nodes_hit);
    /*=============================================*/
    /* No node released case */
    printf("\t==========\n");
    printf("\tExperiment: Reverse Tracing Scheme Method\n");
    normal_result_case_1=iterative_reverse_tracing_scheme(a_mesh, NULL );
    /*=============================================*/
    /* Other cases---Furthest-node-detachment method */
    set_up_furthest_node(a_mesh);
        printf("\t==========\n");
        printf("\tExperiment: Reverse Tracing Scheme Method\n");
    /* Check the contact node inside the rectangle. */
    /* All contact nodes are inside the rectabgle,  */
    /* furthest_node1_index was set to -1.          */
    if(a_mesh->furthest_node1_index < 0){
        printf("\tAll contact nodes are inside the rectangle.\n");
        printf("\tNo test on the released-node case in this increment.\n");
        normal_result_case_2= -1;
    }
    else{
        a_node=a_mesh->hit_list[a_mesh->furthest_node1_index];
        normal_result_case_2=iterative_reverse_tracing_scheme(a_mesh, a_node );
    }
    /*=============================================*/
    /* Compare the results */
    if((normal_result_case_1==1 && normal_result_case_2==1)
       || (normal_result_case_1==0 && normal_result_case_2==0)){
       if(a_mesh->shape_function_case_1 < a_mesh->shape_function_case_2){
            /* Use case 1*/
            copy_case1_to_system(a_mesh);
            best_function=a_mesh->shape_function_case_1;
       }
       else{
            /* Use case 2*/
            node_to_be_released=1;
            copy_case2_to_system(a_mesh);
            best_function=a_mesh->shape_function_case_2;
       }
    }
    else if(normal_result_case_1==-1 && normal_result_case_2==-1){
        printf("PRPBLEM: Both cases encountered a problem in the first
        calculation!\n");
        printf("         or no-node released encountered a problem in
        the first calculation\n");
        printf("         as all contact nodes inside the rectangle.\n");
```

```
        cerror("PROBELM in Reverse Tracing Scheme !!");
    }
    else if(normal_result_case_1 > normal_result_case_2){
         /* one_release encountered a problem in first iteration */
         /* Use case 1: no release*/
         copy_case1_to_system(a_mesh);
         best_function=a_mesh->shape_function_case_1;
    }
    else /* so use the other case, i.e. one release */
    {    /* no_release encountered a problem in first iteration */
         /* Use case 2: one relaese*/
         node_to_be_released=1;
         copy_case2_to_system(a_mesh);
         best_function=a_mesh->shape_function_case_2;
    }
   /* Clean the spare_reverse_case_1_2 */
   remove_spare_reverse_case_1_2(a_mesh);
   /* Move mesh and top die upwards */
   /* move_point_for_RTS(a_mesh, -a_mesh->invstep);*/
   move_die_by(&a_mesh->top_die, +a_mesh->invstep);
   correct_hit_nodes(a_mesh);
   update_matrix_unknowns(a_mesh);
   a_mesh->done_distance += a_mesh->invstep;
   /*=============================================*/
    printf("\t----------------------------\n");
    printf("\tBest shape function was %.5e.\n", best_function);
    if (node_to_be_released>0){printf("\tNode released = %i.\n",
       a_node->marker);
    }
    else {printf("\tNo nodes were released.\n");
    }
    /*Reverse data dump*/
    if (node_to_be_released>0){
        fprintf(a_mesh->outfile, "%i ", a_node->marker);
    } else {
        fprintf(a_mesh->outfile, "- ");
    }

}

/*************************************************************/
int
iterative_reverse_tracing_scheme(struct mesh *a_mesh, struct node *a_node)
{
    int  i;
    int max_num_iteration=20;
    real shape_convergence=0.005; /* MAY be changed */
    real shape_error=BIG;
    int num_iteration=0;
    int stop_iteration=0;
    int normal=1;
    /* normal<=0, for the calculation encounter a problem.    */
    /* normal=0, after firt calculation encounters a problem, */
    /* use the solution so far for new reversed shape         */
    /* normal=-1,  first calculation encounters a problem,    */
    /* no new reversed shape. */
    real shape_function;
    real ctime=a_mesh->invstep;
    /* RECORD X */
    copy_to_spare_reverse(a_mesh);
    copy_to_spare_hit_list(a_mesh);
    /* Check no node release */
    if(a_node != NULL){
        printf("\tRelease node = %i.\n", a_node->marker);
        release_node(a_node);
        /* Remove the node from hit_list for correct_hit_nodes, */
        /* therefore, correct_hit_nodes() will work correctly.  */
        for(i=1;i<=a_mesh->nodes_hit;i++){
            if(a_node->marker == a_mesh->hit_list[i]->marker){
```

```
            remove_node_from_hit_list(a_mesh,i);
        }
    }
    /* Update the matrix index of unknowns and other information */
    update_matrix_unknowns(a_mesh);
}
else{
    printf("\tNo node released.\n");
}
    printf("\t==========\n");

/* Solve FORWARD */
if (solve_squash(a_mesh)){
    /* To form a new reversed shape X1 */
    move_point_for_RTS(a_mesh, -ctime);
    /* Also reverse top die */
    move_die_by(&a_mesh->top_die, +ctime);
    correct_hit_nodes(a_mesh);
    update_matrix_unknowns(a_mesh);
}
else{
      printf("\t Problem. normal=-1\n");
      /* normal=-1; */
      normal=1;
}
shape_error=BIG;
num_iteration=0;
stop_iteration=0;
while(normal==1 && stop_iteration==0){
    /* Solve FORWARD New X1 -> X0_new*/
    if (solve_squash(a_mesh)){
        move_point_for_RTS(a_mesh, +ctime);
        /* Compare X1=X0_new */
        shape_error=compare_shape_error(a_mesh);
    }
    else{
         printf("\t Problem_RTS. normal=0\n");
         normal=0;
    }
    /* Compare X1=X0_new */
    if( shape_error >= shape_convergence ){
    /* YES */
        ++num_iteration;
        if( num_iteration <= max_num_iteration){
            /* X0_new=X0_old; */
            /* Resume the old shape only, not velocity */
            resume_point_from_spare_reverse(a_mesh);
            move_point_for_RTS(a_mesh, -ctime);
            correct_hit_nodes(a_mesh);
            update_matrix_unknowns(a_mesh);
        }
        else{
          printf("\t Problem: Exceed Max iteration !!.\n");
          normal=0;
        }
    }
    /* NO */
    else{
       /* Very  close: Next BCs */
       stop_iteration=1;
       resume_point_from_spare_reverse(a_mesh);
       move_point_for_RTS(a_mesh, -ctime);
       /* correct_hit_nodes(a_mesh); */
       /* update_matrix_unknowns(a_mesh); */
    }
}
if(normal>=0){ /* unussal case */
    /* the other case normal=-1,
```

```
            which no any calculation obtained
            so that there is no reversed shape */
        /*normal=0 has a problem after the first calculation,
          also recored the result */
        shape_function=do_shape_function(a_mesh);
        /* Record the results; */
        if(a_node != NULL){
            /* For one-node release */
            copy_to_spare_reverse_case_2(a_mesh);
            a_mesh->shape_function_case_2=shape_function;
        }
        else{
            /* For no-node release */
            copy_to_spare_reverse_case_1(a_mesh);
            a_mesh->shape_function_case_1=shape_function;
        }
    }
    /* Get back the information for the next new BCs*/
    move_die_by(&a_mesh->top_die, -ctime);
    retrieve_spare_reverse(a_mesh);
    retrieve_spare_hit_list(a_mesh);
    return(normal);
}

/*************************************************************
 * Solve a fixed distance for different Boundary Conditions
 * and return a shape function.
 * "a_node" is going to be released.
 * If a_node == NULL, no node will be released.
 *************************************************************/
real
solve_reverse_in_fixed_distance_BCs(struct mesh *a_mesh, struct node *a_node )
{
    real previous_done_distance=a_mesh->done_distance;
    real die_moved_dist=0.0;
    real shape_function;
    int  moved_die=0;
    int  i;
    int step_inside=0;
    int step_inside_max= 1;
    int encounter_problem = 0;
    /* A flag to show the slover encounters a problem before
     * it reaches the a_mesh->invstep. Mode 0 for No problem.
     */
    a_mesh->step_done_distance= 0.0;
    copy_to_spare_reverse(a_mesh);
    copy_to_spare_hit_list(a_mesh);
    /* Check no node release */
    if(a_node !=NULL){
        printf("\tTEST      : Node %i released.\n", a_node->marker);
        release_node(a_node);
        /* Remove the node from hit_list for correct_hit_nodes */
        /* therefore, correct_hit_nodes() will work correctly  */
        /* copy_to_spare_hit_list() wil put it back            */
        for(i=1;i<=a_mesh->nodes_hit;i++){
            if(a_node->marker == a_mesh->hit_list[i]->marker){
                remove_node_from_hit_list(a_mesh,i);
            }
        }
        /* Update the matrix index of unknowns and other information */
        update_matrix_unknowns(a_mesh);
    }
    else{
        printf("\tTEST      : No node released\n");
    }

    /*Solve this system*/
    previous_done_distance=a_mesh->done_distance;
```

```
a_mesh->step_done_distance=0.0;
die_moved_dist=0.0;
if (solve_squash(a_mesh)){
    ++step_inside;
    printf("\tInside step = %i has finished.\n", step_inside);
    adjust_time_for_reverse_moment(a_mesh, a_mesh->step_done_distance);
    correct_hit_nodes(a_mesh);
    /* Update the matrix index of unknowns and other information */
    update_matrix_unknowns(a_mesh);
    moved_die=1;
    a_mesh->step_done_distance = a_mesh->done_distance -
    previous_done_distance;
    die_moved_dist +=  a_mesh->ctime;
    while((a_mesh->step_done_distance < a_mesh->invstep) &&
    (step_inside < step_inside_max)){
        if (solve_squash(a_mesh)){
            adjust_time_for_reverse_moment(a_mesh,
            a_mesh->step_done_distance);
            correct_hit_nodes(a_mesh);
            /* Update the matrix index of unknowns and other information */
            update_matrix_unknowns(a_mesh);
            a_mesh->step_done_distance = a_mesh->done_distance -
            previous_done_distance;
            die_moved_dist +=  a_mesh->ctime;
            ++step_inside;
            printf("\tInside step = %i has finished.\n", step_inside);
            printf("\tMove mesh to finsh the rest increment distance!!\n");
        }
        else {
            /* Couldn't solve the system - Covergence problem
               it's not good */
            encounter_problem = 1;
            break;
        }
    }

   /* Check if the step_indide in this increment has
    * reached the step_inside_max.
    * If it has reached,  the mesh is moved by the velocity solved
    * so far to finish the rest of the increment,
    * in order to save the CPU time.
    */
    if(step_inside >= step_inside_max){
        /****
        printf("\tWarning : Step inside has reached step_inside_max =
        %i.\n", step_inside_max);
        printf("\t          Solve the system so far.\n");
        printf("\t          Move mesh to finsh the rest increment
        distance.\n");
        ****/
        printf("\tEnforce: Move mesh to finsh the rest distance
        for this step!!\n");
        a_mesh->ctime=a_mesh->invstep - a_mesh->step_done_distance;
        die_moved_dist +=  a_mesh->ctime;
        move_mesh(a_mesh); /* including move die */
        correct_hit_nodes(a_mesh);
        /* Calculate the  done_distance */
        /* Not needed in this function  */
        /* a_mesh->done_distance= previous_done_distance+a_mesh->invstep;*/
    }
    else{
        printf("\tNormal: Move mesh.\n");
    }

}/* if */
else {
    /* Couldn't solve the system - it's not good */
    encounter_problem = 1;
```

```
        moved_die=0;
    }

    /* Get the shape function, the system can solve so far. */
    if(encounter_problem){
        /*Couldn't solve the system - it's not good*/
        printf("\tEncounter a convergence problem during calculation.\n");
        printf("\t**** shape_function=BIG ******\n");
        shape_function=BIG;
    }
    else{
        shape_function=do_shape_function(a_mesh);
    }
    /* Get die back */
    if (moved_die) move_die_by(&a_mesh->top_die, -die_moved_dist);
    /* Get back the information for the next go*/
    retrieve_spare_reverse(a_mesh);
    retrieve_spare_hit_list(a_mesh);
    /* Remove the spare
     * This part has written inside retrieve_spare()
     * remove_spare(a_mesh) is not necessary.
     */
    /* Check the system index again because in some case
     * the index and contact-flash condition is destoried.
     * e.g. Couldn't solve the system then "break" in while loop,
     * in this case mark_interior()...three functions have to run once.
     * To Update the matrix index of unknowns and other information
     */
     correct_hit_nodes(a_mesh);
     update_matrix_unknowns(a_mesh);
    return shape_function;
}

/*************************************************************
 * Solve a reverse simulation in a fixed distance
 * after the BCs has been given.
 *
 * This fuction is similar to the function
 * solve_reverse_in_fixed_distance_BCs().
 * The difference is that there is no copy_spare()
 * and retrieve() functions in this function.
 * It is possible to combine these two function into one.
 * However, it will be better to seperate during the development.
 * The fix-distance increment is set by a_mesh->invstep.
 *************************************************************/
void
solve_reverse_in_fixed_distance(struct mesh *a_mesh, struct node *a_node){
    real previous_done_distance=a_mesh->done_distance;
    /* real step_done_distance=0.0; */
    int  i;
    int step_inside=0;
    int step_inside_max= 1;
    printf("\n\t----------\n");
    printf("\tDo the best Experiment again.\n");
    printf("\t----------\n");
    /* Check no node release */
    if(a_node != NULL){
        printf("\tRelease node = %i.\n", a_node->marker);
        release_node(a_node);
        /* Remove the node from hit_list for correct_hit_nodes, */
        /* therefore, correct_hit_nodes() will work correctly.  */
        for(i=1;i<=a_mesh->nodes_hit;i++){
            if(a_node->marker == a_mesh->hit_list[i]->marker){
                remove_node_from_hit_list(a_mesh,i);
            }
        }
        /* Update the matrix index of unknowns and other information */
        update_matrix_unknowns(a_mesh);
    }
    else{
```

```
    printf("\tNo node released.\n");
}
    printf("\t==========\n");
/* Solve this system */
previous_done_distance=a_mesh->done_distance;
a_mesh->step_done_distance=0.0;
if (solve_squash(a_mesh)){
    ++step_inside;
    printf("\tInside step = %i has finished.\n", step_inside);
    adjust_time_for_reverse_moment(a_mesh, a_mesh->step_done_distance);
    correct_hit_nodes(a_mesh);
    /* Update the matrix index of unknowns and other information */
    update_matrix_unknowns(a_mesh);
    a_mesh->step_done_distance = a_mesh->done_distance -
    previous_done_distance;
    while( a_mesh->step_done_distance < a_mesh->invstep &&
    (step_inside < step_inside_max)){
        if (solve_squash(a_mesh)){
            ++step_inside;
            printf("\tInside step = %i has finished.\n", step_inside);
            printf("\tMove mesh to finsh the rest increment distance!!\n");
            adjust_time_for_reverse_moment(a_mesh,
            a_mesh->step_done_distance);
            correct_hit_nodes(a_mesh);
            /* Update the matrix index of unknowns and other information */
            update_matrix_unknowns(a_mesh);
            a_mesh->step_done_distance = a_mesh->done_distance -
            previous_done_distance;
        }
        else {
            /* Encounter a problem. Solve the system so far.*/
            printf("\tEncounter a convergence problem AFTER
            the 1st calculation.\n");
            printf("\tSolve the system so far.\n");
            break;
        }
    } /* while */
    /* Check if the step_indide in this increment
     * has reached the max. = 20
     * If it has reached, the mesh is moved by the velocity
     * solved so far to finish the rest of the increment,
     * in order to save the CPU time.
     */
    if(step_inside >= step_inside_max){
        /****
        printf("\tWarning : Step inside has reached step_inside_max =
        %i.\n",step_inside_max );
        printf("\t          Solve the system so far.\n");
        printf("\t          Move mesh to finsh the rest increment
        distance.\n");
        ****/
        printf("\tEnforce: Move mesh to finsh the rest distance
        for this step!!\n");
        a_mesh->ctime=a_mesh->invstep - a_mesh->step_done_distance;
        move_mesh(a_mesh); /* including move die */
        correct_hit_nodes(a_mesh);
        /* Calculate the done_distance */
        a_mesh->done_distance= previous_done_distance + a_mesh->invstep;
    }
    else{
        printf("\tNormal: Move mesh.\n");
    }
}
else{
    /* Couldn't solve system in the first calculation,
     * then enforce mesh to move a small distance (time)
```

```
         * At the moment, a_mesh->step_done_distance=0,
         * adjust_time_for_reverse_moment() will enforce a small distance.
         */
         printf("\tEncounter a convergence IN the 1st calculation.\n");
         printf("\tMove the mesh by using current velocity field.\n");
         adjust_time_for_reverse_moment(a_mesh, a_mesh->step_done_distance);
        /* Enforce to move the mesh by current velocity field.
         * a_mesh->ctime=a_mesh->invstep;
         * adjust_time_for_reverse_moment(a_mesh, a_mesh->step_done_distance);
         */
    }
    /* Correct the hit nodes on the die.
     * Update the matrix index of unknowns and other information.
     * These two functions are performed for providing a correct
     * the information for the next calculation.
     */
    correct_hit_nodes(a_mesh);
    update_matrix_unknowns(a_mesh);
}

/*********************************************************
 * Furthest-node-detachment
 * This function looks for the furthest node.
 *********************************************************/
void
set_up_furthest_node(struct mesh *a_mesh){
    int i, j;
    int  furthest_node_index;
    /* the position of the hit-list-vectory, a_mesh->hit_list[furthest_node]*/
    real distance, furthest_distance;
    real a_nx,  a_ny;
    real x_right, x_left, y_bot, y_top;
    /* rectangle boundary at the same centre with the mesh */
    /* struct node *a_n; */
    real centre[KDIM];
    furthest_node_index=-1;
    /* -1 means all contact nodes are inside the rectangle */
    furthest_distance=0.0;
    distance=0.0;

    /* Caculate the 2nd moment of the area */
    moment_x_y(a_mesh);
    /* Find the centre of gravity of the mesh.
     * This part has caculated in function:moment_x_y().
     * assign to  centre[j].
     * For the symmetry on x=0, the centre[0]=0.
     */
    for(j=0;j<KDIM;j++) centre[j]=a_mesh->centre[j];

    /* do function, because return a value */
    /* set up rectangle_width and rectangle_height */
    if(rectangle_min_distance(a_mesh))
    x_right=centre[0] + a_mesh->rectangle_width/2.0;
    x_left =centre[0] - a_mesh->rectangle_width/2.0;
    y_top  =centre[1] + a_mesh->rectangle_height/2.0;
    y_bot  =centre[1] - a_mesh->rectangle_height/2.0;

    for (i=1;i<=a_mesh->nodes_hit;i++){
        a_nx=a_mesh->hit_list[i]->point[0];
        a_ny=a_mesh->hit_list[i]->point[1];
        /* consider where is the location of the node */
        if ( a_nx > x_right  && (a_ny > y_bot && a_ny < y_top) ){
            distance=a_nx-x_right;
        }
        else if(a_nx > x_right && a_ny > y_top){
            distance=sqrt( (a_nx-x_right)*(a_nx-x_right)
                           +(a_ny-y_top)*(a_ny-y_top));
        }
        else if( (a_nx < x_right && a_nx >x_left) && (a_ny > y_top)){
            distance=a_ny-y_top;
```

```
        }
        else if( a_nx < x_left  && a_ny > y_top){
            distance=sqrt( (a_nx-x_left)*(a_nx-x_left)
                           +(a_ny-y_top)*(a_ny-y_top));
        }
        else if ( a_nx < x_left  && (a_ny > y_bot && a_ny < y_top) ){
            distance=x_left-a_nx;
        }
        else if( a_nx < x_left  && a_ny < y_bot){
            distance=sqrt( (a_nx-x_left)*(a_nx-x_left)
                           +(a_ny-y_bot)*(a_ny-y_bot));
        }
        else if( (a_nx < x_right && a_nx >x_left) && (a_ny < y_bot)){
            distance=y_bot-a_ny;
        }
        else if(a_nx > x_right && a_ny < y_bot){
            distance=sqrt( (a_nx-x_right)*(a_nx-x_right)
                           +(a_ny-y_bot)*(a_ny-y_bot));
        }
        else {distance = 0.0;} /* the node inside the rectangle */

        /* Check the furthest node */
        /* First Check special case for Horizontal line */
        if(fabs(distance - furthest_distance) < 0.000001){ /* very close */
            /* If the distance is very close,
             * then select the point with larger x-coordinate.
             */
            if(furthest_node_index > -1){
             /* mode -1 : without furthest_node_index*/
                if( a_nx > a_mesh->hit_list[furthest_node_index]->point[0]){
                    furthest_distance = distance;
                    furthest_node_index = i;
                }
            }
        }
        /* Check normal case */
        else if(distance > furthest_distance){
           furthest_distance = distance;
           furthest_node_index = i;
        }
    }
    a_mesh->furthest_node1_index = furthest_node_index;
    return;
}

/*************************************************************/
real
compare_shape_error(struct mesh *a_mesh){
    real e_norm=0;
    int i, j;
    for (i=1;i<=a_mesh->num_node;i++){
        struct node *a = &a_mesh->nodev_spare_reverse[i];
        struct node *b = &a_mesh->nodev[i];
        for(j=0; j<KDIM;++j){
            e_norm=e_norm+ (a->point[j] - b->point[j])*(a->point[j] -
            b->point[j]);
        }
    }
    e_norm=sqrt(e_norm)/a_mesh->num_node;
    printf("\tshape_error=%.8f\n",e_norm);
    return e_norm;
}

/*************************************************************/
void
resume_point_from_spare_reverse(struct mesh *a_mesh){
    int i, j;
    for (i=1;i<=a_mesh->num_node;i++){
        struct node *a = &a_mesh->nodev_spare_reverse[i];
        struct node *b = &a_mesh->nodev[i];
        for(j=0; j<KDIM;++j){
```

```
            b->point[j] = a->point[j];
        }
    }
}

/*************************************************************
 * A value returned from distance indicates
 * that the node is at the shortest distance to
 * a shell (Prependicular).
 * If the node is a prependicular case,
 * the foot point on the plane is stored in pt[].
 * Otherwise, the node closest distance is returned.
 *************************************************************/
real
shortest_distance_node_shell(struct node *a_node, struct shell
*a_shell, real pt[KDIM])
{
    /* real pt[KDIM];  Point found on the plane Ax+By+Cz+D=0  */
    real coeff[KDIM+1];
    /* Coefficients A, B, C and D for plane Ax+By+Cz+D=0      */
    real dist;
    /* Prependicular Distance from the node to the foot point */
    real area_foot_pt;
    /* Area constructed by the point found and the shell's vertex */
    real m;             /* Calcutation Coefficient */
    real temp;
    int i,j,k;
    /* Initialize(assign) cs_simplex[i][j]=1, at i=j, =0.0 at i<=>j*/
    /* use 3*3 materix, cs_simples[][] was defined in mesh.h */
    for (i=0; i<3; i++){
        for(j=0;j<3;j++){
            if(i==j) {cs_simplex[j+1][i+1]=1.0;}
            else {cs_simplex[j+1][i+1]=0.0;}
        }
    }
    /* Caiculate the coefficients for the plane Ax+By+Cz+D=0 */
    for(k=0;k<=KDIM;k++){
        /* assign cs_simplex[][] form a shell */
        for(i=0;i<KDIM;i++){
            for (j=0; j<KDIM;j++){
                cs_simplex[i+1][j+1]=a_shell->vertex[i]->point[j];
            }
        }
        if (k<KDIM){   /* as k=KDIM, we don't need to change */
            for (j=0; j<KDIM;j++){
                cs_simplex[j+1][k+1]=1.0;
            }
        }
        coeff[k]= determinant_33(cs_simplex);
    }
    coeff[KDIM]=-coeff[KDIM];   /* Coefficients D, for plane Ax+By+Cz+D=0  */

    /* The foot point pt[] of the perpendicular from the node */
    temp=0.0;
    m=0.0;
    for(i=0;i<KDIM;i++){ temp += coeff[i]*coeff[i];}
    for(i=0;i<KDIM;i++){ m += coeff[i]*a_node->point[i];}
    m += coeff[KDIM];
    m /= -temp;
    for(i=0;i<KDIM;i++) pt[i]=m*coeff[i]+a_node->point[i];
    /* Check the point found is on the plane(inside the plane
     * Using the area construct by the foot point fount to check
     *  the foot point is on the plane
     */
    area_foot_pt=0.0;
#if (KDIM==2)
    /* For 2D, check length */
    for(k=0;k<KDIM;k++){
        /* Find the vector between the points */
        for(i=0;i<KDIM;i++){ cs_rhs[i+1]=pt[i]-a_shell->vertex[k]->point[i];}
```

```
                            /* cs_rhs[] was defined in mesh.h */
        temp=0.0;
        for(i=1;i<=KDIM;i++){ temp += (cs_rhs[i]*cs_rhs[i]);}
        area_foot_pt= area_foot_pt + fsqrt(temp);
    }
#endif
#if (KDIM==3)
    for(k=0;k<KDIM;k++){
        /* Make simplex of positional vectors for cross product*/
        /* Determinant */
        for(j=1;j<=KDIM;j++) cs_simplex[1][j]=1.0 ;
            /* Vectors between the foot node and shell's vertex */
            for (i=1;i<KDIM;i++){
                int v=i+k-1;
                if(v>=KDIM) v=0;
                for(j=0;j<KDIM;j++){
                    cs_simplex[i+1][j+1]=
                        pt[j]- a_shell->vertex[v]->point[j];
                }
            }
        /* Get determinant of the simplex */
        /* Then get the one area of the three triangles which
         * constructed by the foot oint
         */
        area_foot_pt += (0.5* fabs(determinant_33(cs_simplex)));
    }
#endif

/*****************************************************************
 * This function calculates the moment of a mesh, Ix and Iy,
 * at the gravity centre.
 * Gauss quadrature is used to caculate the moment of
 * the workpiece.
 *****************************************************************/
void
moment_x_y(struct mesh *a_mesh)
{
   struct element *an_el;
   real m_x,  m_y;
   /* real centre_x_el, centre_y_el; */
   real centre[KDIM];
   real x[3], y[3];
   int i, j;
   real symmetry_factor=1.0;
    /* Symmetry x=0 */
   if (a_mesh->symmetry_x) symmetry_factor*=2.0;
   /* Symmetry y=0 */
   /* if (a_mesh->symmetry_x) symmetry_factor*=2.0; */
   m_x=0.0;
   m_y=0.0;
   /*Find the centre of gravity of the mesh*/
   for (j=0;j<KDIM;j++) centre[j]=0.0;
   for (i=1;i<=a_mesh->num_node;i++){
        for(j=0;j<KDIM;j++) centre[j]+=a_mesh->nodev[i].point[j];
   }
   for(j=0;j<KDIM;j++) centre[j]/=a_mesh->num_node;
   /* Symmetry x=0 */
   if (a_mesh->symmetry_x) centre[0]=0.0;
   /* assign to a_mesh */
   for(j=0;j<KDIM;j++) a_mesh->centre[j]=centre[j];

   for (i=1;i<=a_mesh->num_elem;i++){
        /*
        centre_x_el=0.0;
        centre_y_el=0.0;
        */
        an_el=(&a_mesh->elementv[i]);
            for (j=0;j<KDIMP;j++){  /*j is node number*/
                /*
                centre_x_el += an_el->vertex[j]->point[0];
```

```
                centre_y_el += an_el->vertex[j]->point[1];
                */
            /* The node coordinates based on the centre[j] */
            x[j]=an_el->vertex[j]->point[0] - centre[0];
            y[j]=an_el->vertex[j]->point[1] - centre[1];
            }
            /* Gauss Quadrature. */
            /* The moment of the section area, about the x- and the y-axis */
    m_x=m_x +(   (y[0]+y[1]+4.0*y[2])*(y[0]+y[1]+4.0*y[2])
                +(y[2]+y[0]+4.0*y[1])*(y[2]+y[0]+4.0*y[1])
                +(y[1]+y[2]+4.0*y[0])*(y[1]+y[2]+4.0*y[0])
             )*(1.0/36.0)*2.0*an_el->volume/6.0;
    m_y=m_y +(   (x[0]+x[1]+4.0*x[2])*(x[0]+x[1]+4.0*x[2])
                +(x[2]+x[0]+4.0*x[1])*(x[2]+x[0]+4.0*x[1])
                +(x[1]+x[2]+4.0*x[0])*(x[1]+x[2]+4.0*x[0])
             )*(1.0/36.0)*2.0*an_el->volume/6.0;
   }
   a_mesh->moment_x=m_x*symmetry_factor;
   a_mesh->moment_y=m_y*symmetry_factor;
}

/*****************************************************************
 * Test a point p inside which block
 *****************************************************************/
int
point_in_which_block( real p[2], real right, real left, real top, real bot)
{
    int in_side=0;
    /* i.e. inside the rectangle including on the boundary*/
    int in_top=1;
    int in_left=2;
    int in_bot=3;
    int in_right=4;
    if (p[0] <= right && p[0] >= left && p[1] <= top && p[1] >= bot)
        { return in_side; } /* including on the bundary */
    else if (p[1] >= top)
        { return in_top;}
    else if (p[0] <= left )
        { return in_left;}
    else if (p[1] <= bot)
        { return in_bot;}
    else /* if (p[0] >= right) */
        { return in_right;}
}

/*****************************************************************
 * Return the x-coordinate on intersection of rectangle
 *****************************************************************/
real
intersection_top_or_bot(real y, real x_right, real x_left,
real pc[2], real pp[2])
{
    /* pc: point current, pp:point previous         */
    /* Deal with a special case --- Horizontial line*/
    if (fabs(pc[1]-pp[1]) <= EPSILON){
        if(pc[0] >=x_right || pp[0] >= x_right){
            return x_right;
        }
        else if(pc[0] <=x_left || pp[0] <= x_left){
            return x_left;
        }
        else{
            /* pc and pp are on the top or bot line of inside rectangle */
            /* this case is unlikely to happen  */
            return pc[0]; /* pc for current point */
        }
    }
```

```
    /* return x-xoordinate */
    return (pc[0] + (y-pc[1])*( (pp[0]-pc[0])/(pp[1]-pc[1])));
}

/*****************************************************************
 * Return the y-coordinate on intersection of rectangle
 *****************************************************************/
real
intersection_right_or_left(real x, real y_top, real y_bot,
real pc[2], real pp[2])
{
    /* pc: point current, pp:point previous      */
    /* Deal with a special case --- Vertical line*/
    if (fabs(pc[0]-pp[0]) <= EPSILON){
        if(pc[1] >=y_top || pp[1] >= y_top){
            return y_top;
        }
        else if(pc[1] <=y_bot || pp[1] <= y_bot){
            return y_bot;
        }
        else{
            /* pc and pp are on the right or left line of inside rectangle */
            /* this case is unlikely to happen                              */
            return pc[1]; /* pc for current point */
        }
    }
    /* return y-xoordinate */
    return (pc[1] + (x-pc[0])*( (pp[1]-pc[1])/(pp[0]-pc[0])));
}

/*****************************************************************
 * Calculate the difference between the target shape and the
 * deforming shape.
 *****************************************************************/
real
rectangle_area_difference(struct mesh *a_mesh){
    int i, j;
    real centre[KDIM]; /* centre of the rectangle */
    real x_right, x_left, y_bot, y_top;
    /* rectangle boundary at the same centre with the mesh */
    real b_p[1000][2];
    /* Store the coordinate x and y  for bounding points*/
    int flag[1000];
    /* Maximun number of bounding shells is 999 */
    int total_bounding_point=0;
    struct node *c_node; /* connecting node */

    real p[MAX_PolygonPoints][2];
    int n;
    real pp[2], pc[2];
    /* pp: point previous, pc: point current */
    int pp_pos,  pc_pos;
    /* pc position */
    real outside_area;
    real p_temp[2];
    real symmetry_factor=1.0;
    int in_side=0;
    /* i.e. inside the rectangle including on the boundary*/
    int in_top=1;
    int in_left=2;
    int in_bot=3;
    int in_right=4;
    if (a_mesh->symmetry_x) symmetry_factor*=2;
    /* Calculate the nearest rectangle */
    /* Caculate the 2nd moment of the area */
    moment_x_y(a_mesh);
    /* Find the centre of gravity of the mesh.
     * This part has caculated in function:moment_x_y().
     * assign to  centre[j].
```

```
 * For the symmetry on x=0, the centre[0]=0.
 */
for(j=0;j<KDIM;j++) centre[j]=a_mesh->centre[j];
/* set up rectangle_width and rectangle_height */
if(rectangle_min_distance(a_mesh))
/* do if function, because it returns a value */
x_right=centre[0] + a_mesh->rectangle_width/2.0;
x_left =centre[0] - a_mesh->rectangle_width/2.0;
y_top  =centre[1] + a_mesh->rectangle_height/2.0;
y_bot  =centre[1] - a_mesh->rectangle_height/2.0;

/* printf ("Right-L-T-B % f %f %f %f \n",
   x_right,  x_left,  y_top,  y_bot);*/
a_mesh->rectangle_BottomLeft[0]=x_left;
a_mesh->rectangle_BottomLeft[1]=y_bot;
a_mesh->rectangle_TopRight[0]=x_right;
a_mesh->rectangle_TopRight[1]=y_top;
/* Get the boundary points and put into array bound_point[] */
/* flag[i]=1 means the shell has been marked */
/* Index starts from 1, not 0 */
for(i=0;i <= a_mesh->num_bshell; i++) flag[i]=0;
/* Dealing with the first bounding shell */
total_bounding_point=0;
for (i=0; i<2; i++){
    c_node=a_mesh->bshellv[1].vertex[i];
    /*Write down the points*/
    total_bounding_point++;
    b_p[total_bounding_point][0]=c_node->point[0];
    b_p[total_bounding_point][1]=c_node->point[1];
}
flag[1]=1;
/* Dealing with the resmainders */
while (total_bounding_point < a_mesh->num_bshell){
    for (i=2; i <= a_mesh->num_bshell; i++){
        if (flag[i] != 1){
            struct node *a_node=a_mesh->bshellv[i].vertex[0];
            struct node *b_node=a_mesh->bshellv[i].vertex[1];
            /* for 2D only , 2 nodes formed a shell */
            if (c_node == a_node) {
                c_node = b_node;
                flag[i]=1; /* mark it */
                i=a_mesh->num_bshell+1; /* end of loop */
            }
            else if (c_node == b_node) {
                c_node = a_node;
                flag[i]=1; /* mark it */
                i=a_mesh->num_bshell+1; /* end of loop */
            }
        }
    }
    total_bounding_point++;
    /*Write down the points*/
    b_p[total_bounding_point][0]=c_node->point[0];
    b_p[total_bounding_point][1]=c_node->point[1];
}
/* Make polygons which are outside the rectangle.          */
/* Also calculate the sub-area and total outside area      */
outside_area=0.0;
n=0; /* reset number of points of th epolygon */
for(i=1; i<=total_bounding_point; i++){
    /* Make a polygon                                      */
    /* p[][] index starts from 0 for 1st point of the polygon  */
    /* but  b_p[][] index strats from 1                    */
    if (i==1){
        pp[0]=b_p[total_bounding_point][0];
        pp[1]=b_p[total_bounding_point][1];
        pp_pos=point_in_which_block(pp, x_right,x_left,y_top,y_bot);
```

```
}
else if (i==total_bounding_point){
    pp[0]=b_p[1][0];
    pp[1]=b_p[1][1];
    pp_pos=point_in_which_block(pp, x_right,x_left,y_top,y_bot);
}
pc[0]=b_p[i][0];
pc[1]=b_p[i][1];
pc_pos=point_in_which_block(pc, x_right,x_left,y_top,y_bot);

/**** pc inside ****/
if(pc_pos == in_side ){
    /* test extra point */
    if(pp_pos== in_top){    /* from top to inside */
         p_temp[0]=intersection_top_or_bot(y_top, x_right,
         x_left, pc, pp);
         if(p_temp[0] < x_left){
             p[n][0]=x_left;
             p[n][1]=intersection_right_or_left(x_left,y_top,
             y_bot, pc, pp);
             n++;
         }
         else if(p_temp[0] > x_right){
             p[n][0]=x_right;
             p[n][1]=intersection_right_or_left(x_right,y_top,
             y_bot, pc, pp);
             n++;
        }
        else{
             p[n][0]=p_temp[0];
             p[n][1]=y_top;
             n++;
        }
    }
    else if(pp_pos== in_bot){   /* from bot to inside */
         p_temp[0]=intersection_top_or_bot(y_bot, x_right,
         x_left, pc, pp);
         if(p_temp[0] < x_left){
             p[n][0]=x_left;
             p[n][1]=intersection_right_or_left(x_left,y_top,
             y_bot, pc, pp);
             n++;
         }
         else if(p_temp[0] > x_right){
             p[n][0]=x_right;
             p[n][1]=intersection_right_or_left(x_right,y_top,
             y_bot, pc, pp);
             n++;
         }
         else{
             p[n][0]=p_temp[0];
             p[n][1]=y_bot;
             n++;
         }
    }
    else if(pp_pos== in_left){  /* from left to inside */
            p[n][0]=x_left;
            p[n][1]=intersection_right_or_left(x_left,y_top,
            y_bot, pc, pp);
            n++;
    }
    else if(pp_pos== in_right){ /* from right to inside */
            p[n][0]=x_right;
            p[n][1]=intersection_right_or_left(x_right, y_top,
            y_bot, pc, pp);
            n++;
    }
    /* normal inside case */
    p[n][0]=pc[0];
    p[n][1]=pc[1];
    n++;
```

```
}
/**** pc in_top ****/
else if(pc_pos== in_top){
    if(pp_pos==in_side){        /* from inside to top */
        p_temp[0]=intersection_top_or_bot (y_top, x_right,
        x_left, pc, pp);
        if(p_temp[0] < x_left){         /* inside -- left -- top */
            p[n][0]=x_left;
            p[n][1]=intersection_right_or_left(x_left, y_top,
            y_bot, pc, pp);
            n++;
            p[n][0]=x_left;
            p[n][1]=y_top;
            n++;
         }
         else if(p_temp[0] > x_right){  /* inside -- right -- top */
            p[n][0]=x_right;
            p[n][1]=intersection_right_or_left(x_right, y_top,
            y_bot, pc, pp);
            n++;
            p[n][0]=x_right;
            p[n][1]=y_top;
            n++;
        }
        else{ /* between left and right */
            p[n][0]=p_temp[0];
            p[n][1]=y_top;
            n++;
        }
    }
    else if(pp_pos== in_left){  /*from left to top */
        p_temp[0]=intersection_top_or_bot (y_top, x_right,
        x_left, pc, pp);
        if(p_temp[0] < x_left){ /* left -- top */
            p[n][0]=x_left;
            p[n][1]=y_top;
            n++;
         }
        else{                   /* left -- inside -- top */
            p[n][0]=x_left;
            p[n][1]=intersection_right_or_left(x_left, y_top,
            y_bot, pc, pp);
            n++;
            p[n][0]=p_temp[0];
            p[n][1]=y_top;
            n++;
        }
        /* other case > x_right is unlikely to happen */
    }
    else if(pp_pos== in_right){ /* from right to top */
        p_temp[0]=intersection_top_or_bot (y_top, x_right,
        x_left, pc, pp);
        if(p_temp[0] > x_right){ /* right -- top */
            p[n][0]=x_right;
            p[n][1]=y_top;
            n++;
         }
        else{                   /* right -- inside -- top */
            p[n][0]=x_right;
            p[n][1]=intersection_right_or_left(x_right, y_top,
            y_bot, pc, pp);
            n++;
            p[n][0]=p_temp[0];
            p[n][1]=y_top;
            n++;
        }
        /* other case < x_left is unlikely to happen */
    }
    /* othercase pp== in_bot is unlikely */
}
/**** pc in_bot ****/
```

```
else if(pc_pos== in_bot){
    if(pp_pos==in_side){            /* inside -- bot  */
        p_temp[0]=intersection_top_or_bot (y_bot, x_right,
        x_left, pc, pp);
        if(p_temp[0] < x_left){     /* inside -- left -- bot */
             p[n][0]=x_left;
             p[n][1]=intersection_right_or_left(x_left, y_top,
             y_bot, pc, pp);
             n++;
             p[n][0]=x_left;
             p[n][1]=y_bot;
             n++;
         }
         else if(p_temp[0] > x_right){  /* inside -- right -- bot */
             p[n][0]=x_right;
             p[n][1]=intersection_right_or_left(x_right, y_top,
             y_bot, pc, pp);
             n++;
             p[n][0]=x_right;
             p[n][1]=y_bot;
             n++;
        }
        else{ /* between left and right */
             p[n][0]=p_temp[0];
             p[n][1]=y_bot;
             n++;
        }
    }
    else if(pp_pos==in_left){
        p_temp[0]=intersection_top_or_bot (y_bot, x_right,
        x_left, pc, pp);
        if(p_temp[0] <= x_left){    /* left -- bot */
             p[n][0]=x_left;
             p[n][1]=y_bot;
             n++;
         }
        else{                       /* left -- inside -- bot */
             p[n][0]=x_left;
             p[n][1]=intersection_right_or_left(x_left, y_top,
             y_bot, pc, pp);
             n++;
             p[n][0]=p_temp[0];
             p[n][1]=y_bot;
             n++;
        }
        /* other case > x_right is unlikely to happen */
    }
    else if(pp_pos== in_right){
        p_temp[0]=intersection_top_or_bot (y_bot, x_right,
        x_left, pc, pp);
        if(p_temp[0] >= x_right){    /* right -- bot */
             p[n][0]=x_right;
             p[n][1]=y_bot;
             n++;
         }
        else{                       /* right -- inside -- bot */
             p[n][0]=x_right;
             p[n][1]=intersection_right_or_left(x_right, y_top,
             y_bot, pc, pp);
             n++;
             p[n][0]=p_temp[0];
             p[n][1]=y_bot;
             n++;
        }
        /* other case < x_left is unlikely to happen */
    }
    /* othercase pp== in_bot is unlikely */
}
/**** pc in_left ****/
else if(pc_pos== in_left){
```

```
        if(pp_pos==in_side){        /* inside -- left */
                p[n][0]=x_left;
                p[n][1]=intersection_right_or_left(x_left, y_top,
                y_bot, pc, pp);
                n++;
        }
        else if(pp_pos==in_top){
            p_temp[0]=intersection_top_or_bot (y_top, x_right,
            x_left, pc, pp);
            if(p_temp[0] <= x_left){        /* top -- left */
                 p[n][0]=x_left;
                 p[n][1]=y_top;
                 n++;
             }
            else{                       /* top -- inside -- left */
                 p[n][0]=p_temp[0];
                 p[n][1]=y_top;
                 n++;
                 p[n][0]=x_left;
                 p[n][1]=intersection_right_or_left(x_left, y_top,
                 y_bot, pc, pp);
                 n++;
            }
            /* other case > x_right is unlikely to happen */
        }
        else if(pp_pos==in_bot){
            p_temp[0]=intersection_top_or_bot (y_bot, x_right,
            x_left, pc, pp);
            if(p_temp[0] <= x_left){/* bot -- left */
                 p[n][0]=x_left;
                 p[n][1]=y_bot;
                 n++;
             }
            else{                       /* bot -- inside -- left */
                 p[n][0]=p_temp[0];
                 p[n][1]=y_bot;
                 n++;
                 p[n][0]=x_left;
                 p[n][1]=intersection_right_or_left(x_left, y_top,
                 y_bot, pc, pp);
                 n++;
            }
            /* other case > x_right is unlikely to happen */
        }
    }
    /**** pc in_right ****/
    else if(pc_pos==in_right){
        if(pp_pos==in_side){        /* form inside to right */
                p[n][0]=x_right;
                p[n][1]=intersection_right_or_left(x_right, y_top,
                y_bot, pc, pp);
                n++;
        }
        else if(pp_pos==in_top){    /* from top to right */
            p_temp[0]=intersection_top_or_bot (y_top, x_right,
            x_left, pc, pp);
            if(p_temp[0] >= x_right){   /* top --- right */
                 p[n][0]=x_right;
                 p[n][1]=y_top;
                 n++;
            }
            else{                       /* top -- inside -- right */
                 p[n][0]=p_temp[0];
                 p[n][1]=y_top;
                 n++;
                 p[n][0]=x_right;
                 p[n][1]=intersection_right_or_left(x_right, y_top,
                 y_bot, pc, pp);
                 n++;
            }
            /* other case < x_left is unlikely to happen */
```

```
            }
            else if(pp_pos==in_bot){    /* from bot to right */
                p_temp[0]=intersection_top_or_bot (y_bot, x_right,
                x_left, pc, pp);
                if(p_temp[0] >= x_right){ /* right-- right */
                     p[n][0]=x_right;
                     p[n][1]=y_bot;
                     n++;
                }
                else{                     /* bot -- inside -- right */
                     p[n][0]=p_temp[0];
                     p[n][1]=y_bot;
                     n++;
                     p[n][0]=x_right;
                     p[n][1]=intersection_right_or_left(x_left, y_top,
                     y_bot, pc, pp);
                     n++;
                }
                /* other case x< left is unlikely to happen */
            }
        }
        pp[0]=pc[0];
        pp[1]=pc[1];
        pp_pos=pc_pos;
    }/* for */
    /*  for(i=0; i<n; i++){
        printf("pnt t%i %f %f \n", i, p[i][0],  p[i][1]);
        }
    */
     /* p[][] index start from 0 */
     /* We add n++ more than one, so the valid index in p[][] is n-1.*/
     /* However PolygonArea(p, n), n is for number of points.        */
     /* So that the n is the same required number.                   */
    outside_area= PolygonArea(p, n) * symmetry_factor;
    return  outside_area;
}

/*************************************************************
 * Correct the hit nodes position which are away from the die.
 * Work out the shortest distance to the die,  then
 * change the coordinates of the node to the point on the
 * die shell, and change the attached node->hit to the new
 * shell if it is necessary.
 *************************************************************/
void
correct_hit_nodes(struct mesh *a_mesh)
{
    struct node *a_node;
    /* Record the closest point on the shell to the node */
    real temp_pt[KDIM], new_pt[KDIM];
    real temp, min_distance=BIG;
    struct shell *new_hit_shell;
    int hit_top_or_bot=0;
    /* 1 for top die, 2 for bot die */
    int i, j, k, n;
    int not_on_sym_axis=1;
     /* mode 0 for the node on symmetric axis */
    int num_hit_top=0;
    /* Calculate the number of nodes hit the top and bottom die */
    int num_hit_bot=0;
/*Check only the nodes hit the dies */
/* a_mesh->hit_list */
for (i=1;i<=a_mesh->nodes_hit;i++){
    min_distance=BIG;
    not_on_sym_axis=1;
    /* Reset flag for testing whether the node on the sym. axis */
    a_node=a_mesh->hit_list[i];
    for(n=0; n< KDIM; n++){
        if(a_node->index[n] == -3) not_on_sym_axis=0;
```

```
    }
    if(not_on_sym_axis){ /* Ignore the node on symmetric axis for 2D only */
        /*Test against top die*/
        for (j=1;j<=a_mesh->top_die.num_shell;j++){
            struct shell *a_shell=&(a_mesh->top_die.shellv[j]);
            temp=shortest_distance_node_shell(a_node, a_shell, temp_pt);
            if (temp >= 0.0 && temp < min_distance ){
                min_distance=temp;
                for(k=0;k<KDIM;k++){new_pt[k]=temp_pt[k];}
                new_hit_shell=a_shell;
                hit_top_or_bot=1; /* 1 for top die */
            }
        }
        /*Test against bottom die*/
        for (j=1;j<=a_mesh->bot_die.num_shell;j++){
            struct shell *a_shell=&(a_mesh->bot_die.shellv[j]);
            temp=shortest_distance_node_shell(a_node, a_shell, temp_pt);
            if (temp >= 0.0 && temp < min_distance ){
                min_distance=temp;
                for(k=0;k<KDIM;k++){new_pt[k]=temp_pt[k];}
                new_hit_shell=a_shell;
                hit_top_or_bot=2; /* 2 for bottom die */
            }
        }
        /* Correct the node by using the new coordinate and hit_shell */
        if (min_distance>=BIG){
           cerror("ERROR:correct_hit_nodes(): Correct the node position.");
        }
        for(j=0; j< KDIM; j++) a_node->point[j]=new_pt[j];

        /* See if the matrix must be changed,
         * the node contacts the different shell.
         */
        if (a_node->hit != new_hit_shell){
            a_node->hit=new_hit_shell; /* change the hit shell */
            for(i=0;i<KDIM;i++){
                if (a_node->index[n] != -3){
                    a_node->index[i]=0;
                }
                /* Allow make_interior() to allocate a new index
                 * execpt a node on symmetric axis
                 */
            }
        }
        /* Chang, 20_Aug_1997       */
        /* Change other information */
        if ( hit_top_or_bot==1){
            /* a_node->ctime_top=0.0; */
            ++num_hit_top;
        }
        else if ( hit_top_or_bot==2){
            /* a_node->ctime_bot=0.0; */
            ++num_hit_bot;
        }
        index_node(a_mesh, a_node); /* This function is necessary ?? */
    }
}

 /* Special check for nodes on symmetric lines
  * Only consider for 2D cases
  * Because this function is only called by
  * reverse functions in 2D version
  */
  for (i=1;i<=a_mesh->num_node;i++){
      if (a_mesh->nodev[i].is_outside == 2){
              a_mesh->nodev[i].point[0]=0.0;
              a_mesh->nodev[i].velocity[0]=0.0;
      }
  }
```

```
    /* Assign the number of hit nodes to a_mesh.
     * These two values are used to end the simulation.
     */
     a_mesh->num_nodes_hit_top = num_hit_top;
     a_mesh->num_nodes_hit_bot = num_hit_bot;
}
```

A-3.2 Key Routines for Improvement on TEUBA

```
/*****************************************************
 * Reduce Time Step for a better guess of velocity.
 * The CTIME may be too large, which may case the
 * guess of the velocity from the previous increment
 * is too far for the current increment.
 * This function is used to deal with this situation.
 *****************************************************/
void
reduce_time_step(struct mesh *a_mesh,  int reduce_inc){
    int i;
    real d=1.0;
    /* Recover the old position and velocity */
    retrieve_spare(a_mesh);
    copy_to_spare(a_mesh);   /* for next time use*/
    for (i=1; i<=reduce_inc; i++){
        d *= 0.5;
    }
    a_mesh->ctime = (1.0 - d) * (-a_mesh->ctime_record);
    printf("\tCTIME_record= %f\n", a_mesh->ctime_record);
    printf("\tCTIME (Move back by)= %f\n", a_mesh->ctime);
    move_mesh(a_mesh);
    update_matrix_unknowns(a_mesh);
    if(a_mesh->reverse){
        a_mesh->step_done_distance +=a_mesh->ctime;
        /* a_mesh->ctime is negative now. done_distance move back*/
    }
}

/*****************************************************
 * Update the system matrix index of unknowns.
 * The index of matrix and contact conditions
 * are possibly changed, therefor it is necessary
 * to update the information.
 *****************************************************/
void
update_matrix_unknowns(struct mesh *a_mesh){
    /* Check the zero velocity zone flage index mode == -6.
     * Resume the zero velocity zone and allow an index
     * for a velocity component.
     */
    if (a_mesh->zero_velocity_zone_exist){
        resume_zero_velocity_zone(a_mesh);
    }
    else{
       /* the following four functions also inside
        * resume_zero_velocity_zone().
        *
        * For running simulations with Shape function >= 3,
        * the contact condition does not be changed
        * if there is no detachment.
        */
        if (!(a_mesh->reverse & a_mesh->shape_function >= 3)){
            connect_mesh(a_mesh, a_mesh->ctime);
        }
        /*Some things have gone missing from the matrices - so renumber */
        mark_interior(a_mesh);
        a_mesh->matband=bandwidth(a_mesh);
```

```
        if (!(a_mesh->sticking_friction)){
            /* After move mesh, the node contacting
             * to a shell could be changed.
             */
            calculate_flush_contact(a_mesh);
        }
    }
}

/*****************************************************
 * This function is called to deal with the zero
 * energy element in the workpiece.
 * The function checks the energy of each element
 * to see if there is an energy of an element close
 * to zero.
 *****************************************************/
void
zero_energy_element(struct mesh *a_mesh){
    int i, a, b;
    real small_energy_value = 1.0E-4;
    /* Is the value small enough? */
    int check_matrix_index=0;
    a_mesh->zero_velocity_zone_exist=0;
    for(i=1; i<= a_mesh->num_elem; i++){
        struct element *an_el= &a_mesh->elementv[i];
        real energy=an_el->energy;
        if( energy <= small_energy_value){
            /* Mark components of nodes with zero energy and
             * thence take them out of the system eqs.
             */
            for (a=0;a<KDIM;a++){
                for (b=0;b<KDIMP;b++){
                    if ((an_el->vertex[b]->index[a])>0){
                        /* Only check and mark a normal component,
                         * except that the node has a special index
                         * such as -1, -2 = fix by die
                         *            -3 is restricted by symmetry.
                         * -6 is for zero_velocity_zone() to
                         * identofy where will be resumed.
                         */
                        an_el->vertex[b]->index[a]=-6;
                        /* zero energy component */
                        an_el->vertex[b]->velocity[a] =0.0;
                        /* fix the velocity */
                        check_matrix_index=1;
                        /* If the node index is changed,
                         * it is needed to change the matrix index
                         * in system equations.
                         */
                    }
                }
            }
        }
    }
    if (check_matrix_index){
        a_mesh->zero_velocity_zone_exist=1;
        /* Flag for existing zero_velocity_zone.
         * This assignment is possible repeated,
         * if previous trement had been done once.
         */
        mark_interior(a_mesh);
        a_mesh->matband=bandwidth(a_mesh);
    }
}

/*****************************************************************
 * Contact Tolerance Calculation
 * The tolerance depends on the dimension of the mesh model
 * For a small size mesh, the contact_tolerance is
```

```
 * set as a smaller value.
 * The function uses the first elemnet node1 and node2
 * to calculate the tolerance of contact for setting the
 * contact conditions.
 ************************************************************/
void
calculate_contact_tolerance(struct mesh *a_mesh)
{
    struct node *a_node=a_mesh->elementv[1].vertex[0];
    struct node *b_node=a_mesh->elementv[1].vertex[1];
    real p1[KDIM], p2[KDIM];
    int i;
    real sq_distance=0.0;
    for(i=0; i< KDIM; ++i){
        p1[i]=a_node->point[i];
        p2[i]=b_node->point[i];
    }
    /*Square of distance  */
    for(i=0; i< KDIM; ++i){
        sq_distance += (p2[i]-p1[i])*(p2[i]-p1[i]);
    }
    /* Set contact tolerance */
    a_mesh->contact_tolerance = sqrt(sq_distance) * 1.0e-2;
}

//*************************************************************
// Calculate the target shape for OpenInventor model
//*************************************************************
void
caculate_closest_rectangle(struct mesh *a_mesh){
    real x_right, x_left, y_bot, y_top;
    /* rectangle boundary at the same centre with the mesh */
    int j;
    real centre[KDIM];
    moment_x_y(a_mesh);
    // In the function moment_x_y(a_mesh), Centre is caculated
    /*Find the centre of gravity of the mesh*/
    for(j=0;j<KDIM;j++) centre[j]=a_mesh->centre[j];
    /* do function, because return a value */
    /* set up rectangle_width and rectangle_height */
    if(rectangle_min_distance(a_mesh))

    x_right=centre[0] + a_mesh->rectangle_width/2.0;
    if(a_mesh->symmetry_x==1){
        x_left =centre[0];
    }
    else{
        x_left =centre[0] - a_mesh->rectangle_width/2.0;
    }
    y_top  =centre[1] + a_mesh->rectangle_height/2.0;
    y_bot  =centre[1] - a_mesh->rectangle_height/2.0;
    a_mesh->closest_rectangle.cornv[1].point[0]= x_right;
    a_mesh->closest_rectangle.cornv[1].point[1]= y_top;
    a_mesh->closest_rectangle.cornv[2].point[0]= x_left;
    a_mesh->closest_rectangle.cornv[2].point[1]= y_top;
    a_mesh->closest_rectangle.cornv[3].point[0]= x_left;
    a_mesh->closest_rectangle.cornv[3].point[1]= y_bot;
    a_mesh->closest_rectangle.cornv[4].point[0]= x_right;
    a_mesh->closest_rectangle.cornv[4].point[1]= y_bot;
    a_mesh->closest_rectangle.shellv[1].vertex[0]
           =&(a_mesh->closest_rectangle.cornv[1]);
    a_mesh->closest_rectangle.shellv[1].vertex[1]
           =&(a_mesh->closest_rectangle.cornv[2]);
    a_mesh->closest_rectangle.shellv[2].vertex[0]
           =&(a_mesh->closest_rectangle.cornv[2]);
    a_mesh->closest_rectangle.shellv[2].vertex[1]
           =&(a_mesh->closest_rectangle.cornv[3]);
    a_mesh->closest_rectangle.shellv[3].vertex[0]
           =&(a_mesh->closest_rectangle.cornv[3]);
```

```
    a_mesh->closest_rectangle.shellv[3].vertex[1]
           =&(a_mesh->closest_rectangle.cornv[4]);
    a_mesh->closest_rectangle.shellv[4].vertex[0]
           =&(a_mesh->closest_rectangle.cornv[4]);
    a_mesh->closest_rectangle.shellv[4].vertex[1]
           =&(a_mesh->closest_rectangle.cornv[1]);
}

//*****************************************************************
// Draw the target shape
//*****************************************************************
void
draw_closest_rectangle(struct mesh *a_mesh){
    SoSeparator *root=a_mesh->root;
    a_mesh->root_rectangle = new SoSeparator;
    root->addChild(a_mesh->root_rectangle);
    //static long numvertices[1]={KDIM};
    int i;
     /*Create unit indexed arrays for the storage of the arrays*/
    a_mesh->closest_rectangle.shellv=shell_vector(4);
    a_mesh->closest_rectangle.cornv=corner_vector(4);
    a_mesh->closest_rectangle.num_shell=4;
    a_mesh->closest_rectangle.num_corn=4;
    caculate_closest_rectangle(a_mesh);
    for(i=1;i<=a_mesh->closest_rectangle.num_shell;i++){
        SoSeparator *sh=new SoSeparator;
        sh->addChild(rectangle_stuff);
        struct shell *a_shell=&(a_mesh->closest_rectangle.shellv[i]);
        a_shell->coords=new SoCoordinate3;
        calculate_shell(a_shell, 0.999);
        sh->addChild(a_shell->coords);
        SoLineSet *mySet = new SoLineSet;
        sh->addChild(mySet);
        a_mesh->root_rectangle->addChild(sh);
    }
}

//*****************************************************************
// Calculate velocity for OpenInventor model
//*****************************************************************
void
calculate_velocity(node *a_node, float shrink)
{
    float vt[3];
    int i;
    vt[2]=0.0; // special for 2D case
    // for the first point of the vecolity vector
    for (i=0;i<KDIM;i++){
        vt[i] = a_node->point[i];
    }
    a_node->velocities->point.setValues(0, 1, (SbVec3f*) vt);
    // for the second point of the vecolity vector
     for (i=0;i<KDIM;i++){
        vt[i] = a_node->point[i] + shrink*a_node->velocity[i];
    }
    a_node->velocities->point.setValues(1, 1, (SbVec3f*) vt);
 }

//*****************************************************************
// Draw velocity field
//*****************************************************************
void
draw_velocities(struct mesh *a_mesh, SoSeparator *root, float shrink)
{
    int i;
    /* static long numvertices[1]=2; */ /* For a line, two points are enough */
    for(i=1;i<=a_mesh->num_node;i++){
        struct node *a_node=&(a_mesh->nodev[i]);
```

```
        a_node->vel=new SoSeparator;
        a_node->vel->addChild(red_stuff);
        //struct node *a_node=&(a_mesh->nodev[i]);
        a_node->velocities=new SoCoordinate3 ;
        calculate_velocity(a_node, shrink);
        a_node->vel->addChild(a_node->velocities);
        SoLineSet *mySet = new SoLineSet;
        a_node->vel->addChild(mySet);
        root->addChild(a_node->vel);
    }
}

/*************************************************************
 * Record velocity
 *************************************************************/
int
velocity_write(struct mesh *a_mesh)
{
    int i, j;
    int type;
    real *vel;
    /* struct element *an_el; */
    FILE *outfile;
    char filename[Max_filename_length];
    char name[Max_filename_length];
    if (KDIM==2) type=7;
    else if (KDIM==3) type=3;
    else cerror("meshwrite:  What dimension are you working in?");
    sprintf(name, "%s", a_mesh->filename );
    sprintf(filename, "%s.inc%03iv", name,  a_mesh->step);
    if ((outfile=fopen(filename, "w"))==NULL){
        printf("Can't open file %s.\n", filename);
        return(0);
    }
    /*Write down the filename*/
    fprintf(outfile, "    1C%s\n", name);
    /*Write down the node velocity data*/
    fprintf(outfile, "    2C\n");
    for(i=1;i<=a_mesh->num_node;i++){
        /*Number of node*/
        fprintf(outfile, " -1%5i", i);
        /*Velocity of node*/
        vel=a_mesh->nodev[i].velocity;
        for(j=0;j<KDIM;j++){
            /*fegs_form(outfile, pt[j]);*/
            fprintf(outfile, "%12.5E", vel[j]);
        }
        if (KDIM==2) fprintf(outfile, "%12.5E", 0.0);
        fprintf(outfile, "\n");
    }
    fprintf(outfile, " -3\n");
    /*Write down the element data*/
    fprintf(outfile, "    3C\n");
    /*Finish up*/
    fprintf(outfile, " -3\n 9999\n");
    fclose(outfile);
    return(1);
}

/*************************************************************
 * Export bouding shells to a file *.bsh
 *
 * Example:
 *  A triangle with three points, (i.e. three bounding shells)
 *  p1=(0,0) p2=(2,0) p3=(2,1) is save as following structure.
 *  The first part before "2C" is for point data,
 *  and the following part befor "3C" is for line elements.
 * Note: 2C and 3C  are flags for separating points and lines.
 *
```

```
 * 1 0.0 0.0
 * 2 2.0 0.0
 * 3 2.0 1.0
 *  2C
 * 1 1 2
 * 2 2 3
 * 3 3 1
 *  3C
 ************************************************************/
int
bounding_shells_write_2D (struct mesh *a_mesh){
    FILE *outfile;
    char fullname[Max_filename_length];
    int flag[1000]; /* Maximun number of bounding shells is 999 */
    int i;
    int points_foud;
    struct node *c_node; /* connecting node */
    sprintf(fullname, "%s.bsh",a_mesh->filename);
    if ((outfile=fopen(fullname, "w"))==NULL){
        printf("Can't open file %s.\n", a_mesh->filename);
        return(0);
    }
    /* flag[i]=1 means the shell has been marked */
    for(i=0;i <= a_mesh->num_bshell; i++) flag[i]=0;
    points_foud=0;
    /* Dealing with the first bounding shell */
    for (i=0; i<2; i++){
        c_node=a_mesh->bshellv[1].vertex[i];
        points_foud++;
        /*Write down the points*/
        fprintf(outfile,
        "%i %e %e\n",points_foud,c_node->point[0],c_node->point[1]);
    }
        flag[1]=1;
    /* Dealing with the resmainders */
    while (points_foud < a_mesh->num_bshell){
        for (i=2; i <= a_mesh->num_bshell; i++){
            if (flag[i] != 1){
                struct node *a_node=a_mesh->bshellv[i].vertex[0];
                struct node *b_node=a_mesh->bshellv[i].vertex[1];
                /* for 2D only , 2 nodes formed a shell */
                if (c_node == a_node) {
                    c_node = b_node;
                    flag[i]=1; /* mark it */
                    i=a_mesh->num_bshell+1; /* end of loop */
                }
                else if (c_node == b_node) {
                    c_node = a_node;
                    flag[i]=1; /* mark it */
                    i=a_mesh->num_bshell+1; /* end of loop */
                }
            }
        }
        points_foud++;
        /*Write down the points*/
        fprintf(outfile,
        "%i %e %e\n",points_foud,c_node->point[0], c_node->point[1]);
    }
    /*Write down the flag for ending points*/
    fprintf(outfile, " 2C\n");
    /*Write down the bounding shells (lines) */
    for(i=1; i< a_mesh->num_bshell; i++){
        fprintf(outfile,"%i %i %i\n",i , i, i+1);
    }
        /* the final shell */
        fprintf(outfile,
        "%i %i %i\n",a_mesh->num_bshell , a_mesh->num_bshell, 1);
    /*Finish up*/
    fprintf(outfile, " 3C\n");
```

```
    fclose(outfile);
    return(1);
}

#-------------------------------------------------------------
#Program: LoadBSH --- a CADfix (FAM) toolbox script.
#           (BSH stands for Bounding SHells)
#Purpose: Import a file which contains bounding shells
#           information of the workpiece.
#Filename extensions:
#         "bsh" for the imported file
#         (bsh stands for Bounding SHells)
#
#Example: A three bounding shells *.bsh file structure
#         This example has three points (0,0) (10,0) (5,7)
#           and are named point 1, 2 and 3
#         Line 1 is constructed by point 1 and 2.
#         Line 2 is constructed by point 2 point 3 ...etc.
#         "2C" is a flag for ending the points data
#         "3C" is for ending the lines data
#(Start the *.bsh file)
#1  0.0 0.0
#2 10.0 0.0
#3  5.0 7.0
#2C
#1 1 2
#2 2 3
#3 3 1
#3C
#-------------------------------------------------------------

#Get information from FAM
global env famFrame_priv
set modname $famFrame_priv(model)
set ext 01
set exportname [list $modname$ext]

fwDialogueBox .myDialogue -title {TEUBA 2D Mesh}

fwForm .myDialogue.myForm
.myDialogue.myForm add {{Name of billet set} STRING}
.myDialogue.myForm add {{Name of TEUBA mesh Import file *.bsh} STRING}
.myDialogue.myForm set [list {die SET BILL}\
       \ [concat {Root STRING} $modname ]]\

pack .myDialogue.myForm

.myDialogue  action OK {myDialogue:OK}
.myDialogue  action Quit {destroy .myDialogue}
.myDialogue  map

proc myDialogue:OK {} {
  set result [.myDialogue.myForm check]
  if {$result!=0} {
    #Get parameters
    set parameters [.myDialogue.myForm get]
    set billset [lindex $parameters 0]
    set rootnamein [lindex $parameters 1]
    set rootnameout [lindex $parameters 2]
    #
    # It is necessary put something here to check
    # the parameters which are not empity
    #
    #Get information from FAM
    global env famFrame_priv
    set fname $famFrame_priv(modelrootname).fbm
    set dirname $famFrame_priv(dir)
    set modname $famFrame_priv(model)
    #
    #open file to inport the bounding shells
    #
    set file01 [open $dirname/$rootnamein.bsh r]
    #read and plot points of bounding shells
    set pointmarker 1
        #Note 2C for end of points
    while {$pointmarker != "2C" } {
       gets $file01 data
       set pointmarker [lindex $data 0]
```

```
   set pointx [lindex $data 1]
   set pointy [lindex $data 2]
   if {$pointmarker != "2C" } {
       PNT Q$pointmarker $pointx $pointy
   }
}
#read and plot lines of bounding shells
set linemarker 1
    #Note 3C for end of lines
while {$linemarker != "3C" } {
   gets $file01 data
   set linemarker [lindex $data 0]
   set point1 [lindex $data 1]
   set point2 [lindex $data 2]
   if {$linemarker != "3C" } {
       LINE U$linemarker Q$point1 Q$point2
   }
}
#Find three point which are not on the same line
set shapepoint01 1
set shapepoint02 2
set shapepoint03 3
set ok3rdpoint 0
set bigvalue 1.0e+10
seek $file01 0
# 1st point
gets $file01 data
   set shapepoint01 [lindex $data 0]
   set p1x [lindex $data 1]
   set p1y [lindex $data 2]
# 2nd point
gets $file01 data
   set shapepoint02 [lindex $data 0]
   set p2x [lindex $data 1]
   set p2y [lindex $data 2]
   set t1 [expr $p1x-$p2x]
   set t2 [expr $p1y-$p2y]
   if { $t1 == 0} {
        set slope01 [expr $bigvalue]
   } else {
        set slope01 [expr $t2/$t1]
   }
#Look for the 3rd point
while {$ok3rdpoint != 1 } {
   gets $file01 data
   set pointmarker [lindex $data 0]
   set p3x [lindex $data 1]
   set p3y [lindex $data 2]
   set t3 [expr $p1x-$p3x]
   set t4 [expr $p1y-$p3y]
   if { $t3 == 0} {
        set slope02 [expr $bigvalue]
   } else {
        set slope02 [expr $t4/$t3]
   }
   set test [expr $slope01-$slope02]
   if {$test > 0.01 || -$test > 0.01} {
      set shapepoint03 $pointmarker
      set ok3rdpoint 1
   }
}
close $file01
#Make a surface
SHPE X1 Q$shapepoint01 Q$shapepoint02 Q$shapepoint03
SETO $billset
BUILD S ALL X1
SETC $billset
PLOT L
PROC MSTY DELM
```

```
    ELTY $billset TR3
    #Lock the mesh
    #PROC lock on
    destroy .myDialogue
 }
}
#---End

#-------------------------------------------------------------------------
#Program: REMESH2D --- a CADfix (FAM) toolbox program.
#Purpose: Import a file which contains bounding shells information
#         of the workpiece, then re-mesh and save as a new file.
#Filename extensions:
#         "bsh" for the imported file  (bsh stands for bounding shells)
#         "mesh" for the exported file (fr1 file)
#
#Example: A three bounding shells *.bsh file structure
#         This example has three points (0,0) (10,0) (5,7)
#           and are named point 1, 2 and 3
#         Line 1 is constructed by point 1 and 2.
#         Line 2 is constructed by point 2 point 3 ...etc.
#         "2C" is a flag for ending the points data
#         "3C" is for ending the lines data
#(Start the *.bsh file)
#1  0.0 0.0
#2 10.0 0.0
#3  5.0 7.0
#2C
#1 1 2
#2 2 3
#3 3 1
#3C
#-------------------------------------------------------------------------

#Get information from FAM
global env famFrame_priv
set modname $famFrame_priv(model)
set ext 01
set exportname [list $modname$ext]
fwDialogueBox .myDialogue -title {TEUBA 2D Mesh}
fwForm .myDialogue.myForm
.myDialogue.myForm add {{Name of billet set} STRING}
.myDialogue.myForm add {{Name of TEUBA mesh Import file *.bsh} STRING}
.myDialogue.myForm add {{Name of TEUBA mesh Export file *.mesh} STRING}
.myDialogue.myForm set [list {die SET BILL}\
        \ [concat {Root STRING} $modname ]\
        \ [concat {Root STRING} $exportname ]]\
pack .myDialogue.myForm
checkbutton .myDialogue.myCheck  \
  -text "Run FAMnum" \
.myDialogue.myCheck select
pack .myDialogue.myCheck
.myDialogue  action OK {myDialogue:OK $myCheck}
.myDialogue  action Quit {destroy .myDialogue}
.myDialogue  map
proc myDialogue:OK FAMnum {
  set result [.myDialogue.myForm check]
  if {$result!=0} {
    #Get parameters
    set parameters [.myDialogue.myForm get]
    set billset [lindex $parameters 0]
    set rootnamein [lindex $parameters 1]
    set rootnameout [lindex $parameters 2]
    #
    # It is necessary put something here to check
    # the parameters which are not empity
    #
    #Get information from FAM
    global env famFrame_priv
    set fname $famFrame_priv(modelrootname).fbm
    set dirname $famFrame_priv(dir)
    set modname $famFrame_priv(model)
    #
```

```
#open file to inport the bounding shells
#
set file01 [open $dirname/$rootnamein.bsh r]
#read and plot points of bounding shells
set pointmarker 1
    #Note 2C for end of points
while {$pointmarker != "2C" } {
   gets $file01 data
   set pointmarker [lindex $data 0]
   set pointx [lindex $data 1]
   set pointy [lindex $data 2]
   if {$pointmarker != "2C" } {
       PNT Q$pointmarker $pointx $pointy
   }
}
#read and plot lines of bounding shells
set linemarker 1
    #Note 3C for end of lines
while {$linemarker != "3C" } {
   gets $file01 data
   set linemarker [lindex $data 0]
   set point1 [lindex $data 1]
   set point2 [lindex $data 2]
   if {$linemarker != "3C" } {
       LINE U$linemarker Q$point1 Q$point2
   }
}
#Find three point which are not on the same line
set shapepoint01 1
set shapepoint02 2
set shapepoint03 3
set ok3rdpoint 0
set bigvalue 1.0e+10
seek $file01 0
# 1st point
gets $file01 data
   set shapepoint01 [lindex $data 0]
   set p1x [lindex $data 1]
   set p1y [lindex $data 2]
# 2nd point
gets $file01 data
   set shapepoint02 [lindex $data 0]
   set p2x [lindex $data 1]
   set p2y [lindex $data 2]
   set t1 [expr $p1x-$p2x]
   set t2 [expr $p1y-$p2y]
   if { $t1 == 0} {
        set slope01 bigvalue
   } else {
        set slope01 [expr $t2/$t1]
   }
#Look for the 3rd point
while {$ok3rdpoint != 1 } {
   gets $file01 data
   set pointmarker [lindex $data 0]
   set p3x [lindex $data 1]
   set p3y [lindex $data 2]
   set t3 [expr $p1x-$p3x]
   set t4 [expr $p1y-$p3y]
   if { $t3 == 0} {
        set slope02 bigvalue
   } else {
        set slope02 [expr $t4/$t3]
   }
   set test [expr $slope01-$slope02]
   if {$test > 0.01 || -$test > 0.01} {
      set shapepoint03 $pointmarker
      set ok3rdpoint 1
   }
```

```
    }
    close $file01
    #Make a surface
    SHPE X1 Q$shapepoint01 Q$shapepoint02 Q$shapepoint03
    SETO $billset
    BUILD S ALL X1
    SETC $billset
    #Re-mesh
    DIV $billset 1
    PROC MSTY DELM
    ELTY $billset TR3
    MESH $billset
    PLOT M
    PLUS L
    #FAMnum numbering

    if {$FAMnum==1} {
            #Back it up
            BACK =
            fam:send_cmd -nobackspace \"$fname\"
          ####################
          # Save the PX flags
          set shmid $env(PX_SHMID)
          set msqid $env(PX_MSQID)
          set cphnd $env(PX_CPHND)
          # make sure the PX flags are clear
          catch {unset env(PX_SHMID)}
          catch {unset env(PX_MSQID)}
          catch {unset env(PX_CPHND)}
          ####################
            #Do things with FAMnum
            #set inpipe [open |/d2/pd/execs/cadfix200/bin/famnum w+]
            set inpipe [open |$env(FEGS_BIN)/famnum  w+]
            puts $inpipe open
            puts $inpipe $fname
            puts $inpipe rnum
            puts $inpipe back
            puts $inpipe $fname
            puts $inpipe quit
            flush $inpipe
            while {[gets $inpipe line] != -1} {
                fam:send_cmd "** $line"
            }
          ####################
#            close $inpipe
          # Reset the PX flags
          set env(PX_SHMID) $shmid
          set env(PX_MSQID) $msqid
          set env(PX_CPHND) $cphnd
          ####################
      QUIT
      #Pick up the result of FAMnum
      famFrame:openDatabase $fname
    }
    #Save as a FR1 file after re-mesh
    SEND all fr1 \"$dirname/$rootnameout.mesh\"
    #Lock the mesh
    #PROC lock on
    destroy .myDialogue
 }
}
#----End
```